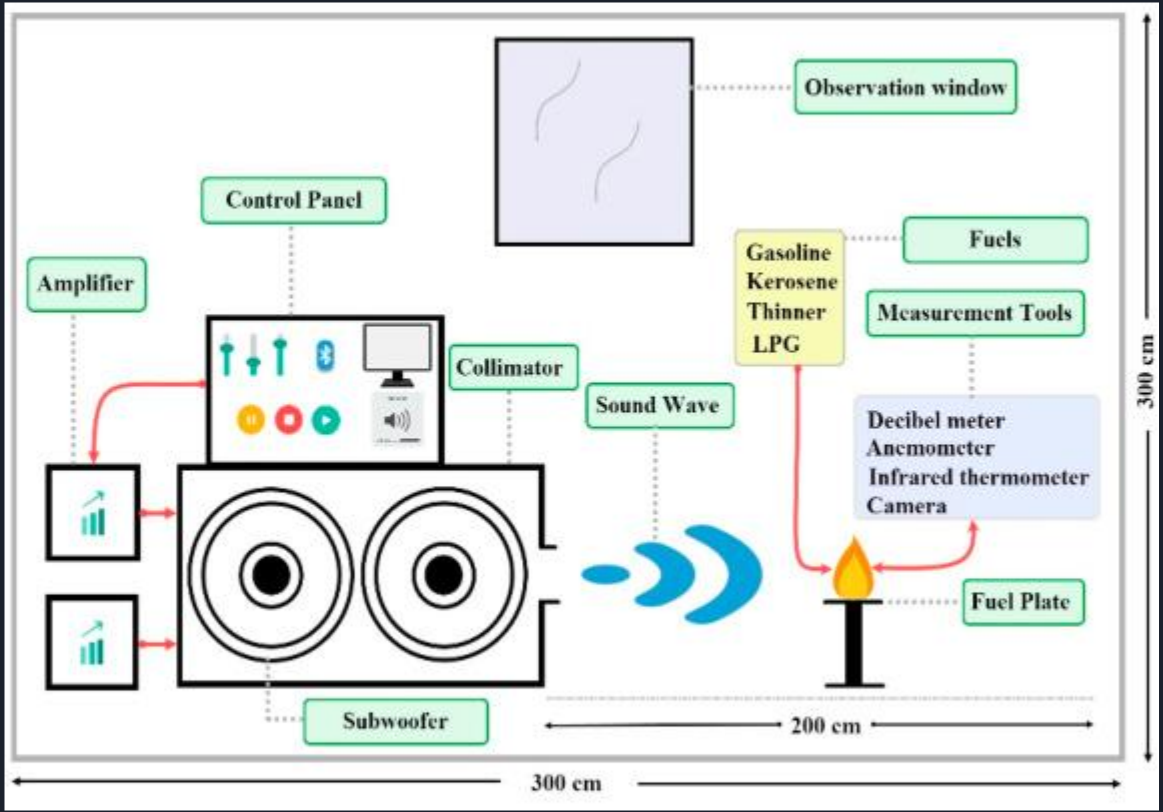
Machine Learning Project

Thomas D and Josh Q

Data Source and Background

- Data is taken from Kaggle: https://www.kaggle.com/datasets/muratkokludataset/acoustic-extinguisher-fire-dataset?resource=download
- The data is from research done on sound waves from speakers putting out fires
- 7 columns and 17,442 rows of data

Study Graphic



Data Overview

Size	Fuel	Distance	Decibel	Airflow	Frequency	Status
1	gasoline	10	96	0	75	0
1	gasoline	10	96	0	72	1
1	gasoline	10	96	2.6	70	1

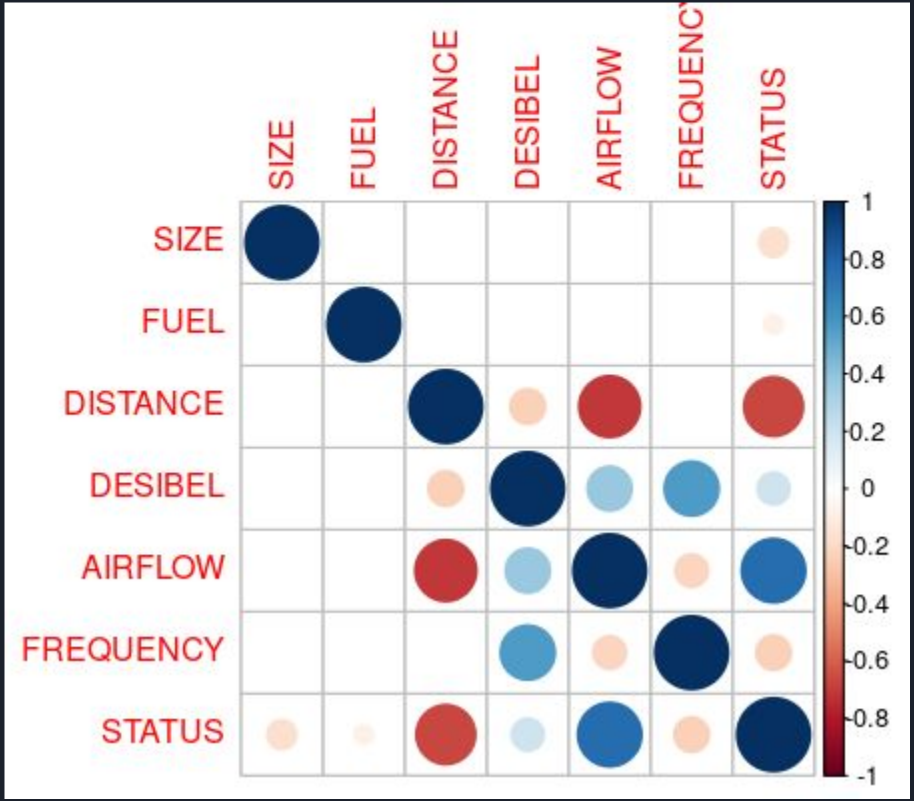
- Fuel: type of fuel used to start fire
- Size: size of canister used to hold fuel
- Distance: Distance of fire from speaker
- Decibel: Volume of speaker
- Airflow: Airflow produced by the speaker
- Frequency: Frequency of sound from the speaker
- Status: Whether or not the fire was put out

Data Overview Cont.

```
      SIZE            FUEL              DISTANCE        DESIBEL
 Min.   :1.000   Length:17442       Min.   : 10    Min.   : 72.00
 1st Qu.:2.000   Class :character   1st Qu.: 50    1st Qu.: 90.00
 Median :3.000   Mode  :character   Median :100    Median : 95.00
 Mean   :3.412                      Mean   :100    Mean   : 96.38
 3rd Qu.:5.000                      3rd Qu.:150    3rd Qu.:104.00
 Max.   :7.000                      Max.   :190    Max.   :113.00
    AIRFLOW         FREQUENCY         STATUS
 Min.   : 0.000   Min.   : 1.00   Min.   :0.0000
 1st Qu.: 3.200   1st Qu.:14.00   1st Qu.:0.0000
 Median : 5.800   Median :27.50   Median :0.0000
 Mean   : 6.976   Mean   :31.61   Mean   :0.4978
 3rd Qu.:11.200   3rd Qu.:47.00   3rd Qu.:1.0000
 Max.   :17.000   Max.   :75.00   Max.   :1.0000
```

- Nearly equal amount of fires put out and fires not put out
- Equal distribution of each size for each fuel type

Correlation Plot



Data Preprocess

- Converted the three fuels into an integers.
- Normalized numeric data using min-max scale to get numbers on a similar scale
- Converted Fuel into categorical data (gasoline, thinner, kerosene, liquid petroleum gas turn into 0,1,2,3).
- Measurement of size was different for LPG than other fuels, so all LPG entries removed from the data set

Size	Fuel	Distance	Decibel	Airflow	Frequency	Status
1	0	10	0.05263158	0.0000000	1.0000000	0
1	0	10	0.05263158	0.0000000	0.9600000	1
1	0	10	0.05263158	0.1529412	0.9333333	1

Training and Testing Data

- Training data was 12,000 randomly selected data points. The testing was then the 3,390 remaining data points
- Seed was set to make the testing and training results reproducible

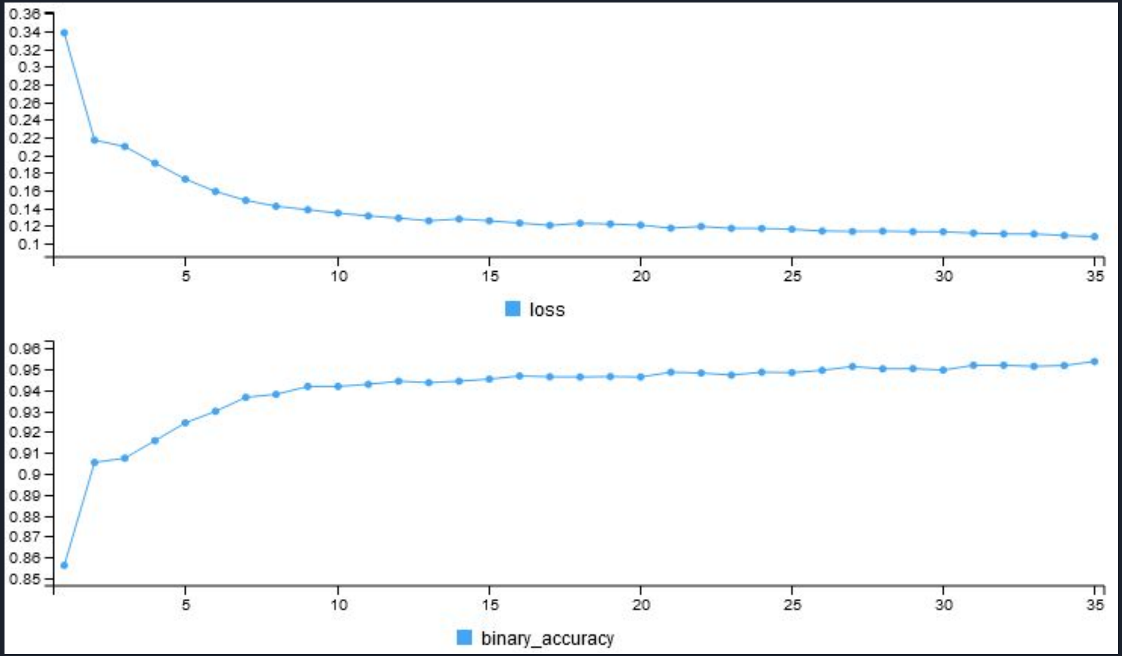
Neural Network Architecture

- 3 layers
- 64 nodes for the two ReLU layers
- 1 node for the output layer, it uses sigmoid
- Loss: binary cross entropy
- Optimizer: adam
- 35 epochs
- Training loss: loss: 0.1119
- Training accuracy: 0.9503
- Testing loss: 0.1438
- Testing accuracy: 0.9342

```
#64 nodes
first_model = keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")


first_model %>% compile(
  loss = "binary_crossentropy",
  metrics = list("binary_accuracy"),
  optimizer = "adam"

)

first_history <- first_model %>%
  fit(
    x = xTr,
    y = yTr, # label is the last column
    epochs = 35,
    validation_data = list(xTest, yTest),
    verbose = 2
  )
```

Run of the model

```
Epoch 1/35
375/375 - 2s - loss: 0.3384 - binary_accuracy: 0.8562 - 2s/epoch - 4ms/step
Epoch 2/35
375/375 - 1s - loss: 0.2167 - binary_accuracy: 0.9054 - 747ms/epoch - 2ms/step
Epoch 3/35
375/375 - 1s - loss: 0.2095 - binary_accuracy: 0.9074 - 740ms/epoch - 2ms/step
Epoch 4/35
375/375 - 1s - loss: 0.1907 - binary_accuracy: 0.9158 - 721ms/epoch - 2ms/step
Epoch 5/35
375/375 - 1s - loss: 0.1726 - binary_accuracy: 0.9243 - 738ms/epoch - 2ms/step
Epoch 6/35
375/375 - 1s - loss: 0.1586 - binary_accuracy: 0.9298 - 762ms/epoch - 2ms/step
Epoch 7/35
375/375 - 1s - loss: 0.1485 - binary_accuracy: 0.9365 - 732ms/epoch - 2ms/step
Epoch 8/35
375/375 - 1s - loss: 0.1418 - binary_accuracy: 0.9379 - 726ms/epoch - 2ms/step
Epoch 9/35
375/375 - 1s - loss: 0.1379 - binary_accuracy: 0.9417 - 764ms/epoch - 2ms/step
Epoch 10/35
375/375 - 1s - loss: 0.1341 - binary_accuracy: 0.9417 - 721ms/epoch - 2ms/step
Epoch 11/35
375/375 - 1s - loss: 0.1309 - binary_accuracy: 0.9427 - 745ms/epoch - 2ms/step
Epoch 12/35
375/375 - 1s - loss: 0.1284 - binary_accuracy: 0.9442 - 740ms/epoch - 2ms/step
Epoch 13/35
375/375 - 1s - loss: 0.1253 - binary_accuracy: 0.9435 - 769ms/epoch - 2ms/step
Epoch 14/35
375/375 - 1s - loss: 0.1274 - binary_accuracy: 0.9442 - 781ms/epoch - 2ms/step
Epoch 15/35
375/375 - 1s - loss: 0.1253 - binary_accuracy: 0.9452 - 729ms/epoch - 2ms/step
Epoch 16/35
375/375 - 1s - loss: 0.1228 - binary_accuracy: 0.9467 - 729ms/epoch - 2ms/step
Epoch 17/35
375/375 - 1s - loss: 0.1202 - binary_accuracy: 0.9463 - 721ms/epoch - 2ms/step
Epoch 18/35
375/375 - 1s - loss: 0.1225 - binary_accuracy: 0.9462 - 722ms/epoch - 2ms/step
Epoch 19/35
375/375 - 1s - loss: 0.1217 - binary_accuracy: 0.9464 - 762ms/epoch - 2ms/step
Epoch 20/35
375/375 - 1s - loss: 0.1205 - binary_accuracy: 0.9462 - 730ms/epoch - 2ms/step
Epoch 21/35
375/375 - 1s - loss: 0.1170 - binary_accuracy: 0.9485 - 727ms/epoch - 2ms/step
Epoch 22/35
375/375 - 1s - loss: 0.1189 - binary_accuracy: 0.9481 - 744ms/epoch - 2ms/step
Epoch 23/35
375/375 - 1s - loss: 0.1168 - binary_accuracy: 0.9471 - 717ms/epoch - 2ms/step
```



Dropout Model

- Example of 3 of the 4 hidden layers having dropout applied to them
- Training loss: 0.0942
- Training accuracy: 0.9603
- Testing loss: 0.0779
- Testing accuracy: 0.9678

```
dropout_model = keras_model_sequential() %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.1) %>%
  layer_dense(units = 128, activation = "relu") %>%
    layer_dropout(0.1) %>%
    layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.1) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

dropout_model %>% compile(
  loss = "binary_crossentropy",
  metrics = list("binary_accuracy"),
  optimizer = "adam"
)

dropout_history <- dropout_model %>%
  fit(
    x = xTr,
    y = yTr,
    epochs = 35,
    validation_data = list(xTest, yTest),
    verbose = 2
  )
```

L2 Regularization Model

- Our model that uses L2 regularization on all 4 of the hidden layers
- Training loss: 0.1405
- Training accuracy: 0.9514
- Testing loss: 0.1300
- Testing accuracy: 0.9581

```r
l2_model = keras_model_sequential() %>%
    layer_dense(units = 128, activation = "relu",
                kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 128, activation = "relu",
                kernel_regularizer = regularizer_l2(l = 0.001)) %>%
      layer_dense(units = 128, activation = "relu",
                kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 128, activation = "relu",
                kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 1, activation = "sigmoid")

l2_model %>% compile(
  loss = "binary_crossentropy",
  metrics = list("binary_accuracy"),
  optimizer = "adam"
)

l2_history <- l2_model %>%
  fit(
    x = xTr,
    y = yTr,
    epochs = 35,
    validation_data = list(xTest, yTest),
    verbose = 2
  )
```

Some tests

- 4 hidden layers, 64 nodes each, 35 epochs
  - loss: 0.0916 - accuracy: 0.9575
- 4 hidden layers, 64 nodes, dropout(0.2), 35 epochs
  - loss: 0.0925 - accuracy: 0.9631
- 4 hidden layers, 128 nodes, L2, 35 epochs
  - loss: 0.1241 - accuracy: 0.9631
- **4 hidden layers, 128 nodes each, 50 epochs**
  - **loss: 0.0711 - accuracy: 0.9714**
- 4 hidden layers, 128 nodes, dropout(0.1), 50 epochs
  - loss: 0.0775 - accuracy: 0.9673
- 6 hidden layers, 128 nodes, L2, 50 epochs
  - loss: 0.1149 - accuracy: 0.9649

```r
first_model = keras_model_sequential() %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 128, activation = "relu") %>%
    layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

first_model %>% compile(
  loss = "binary_crossentropy",
  metrics = list("binary_accuracy"),
  optimizer = "adam"

)

first_history <- first_model %>%
  fit(
    x = xTr,
    y = yTr, # label is the last column
    epochs = 50,
    verbose = 2
  )
```

Final Remarks

- Removing FUEL column of data decreased accuracy about 2%
- Normalizing the data increased accuracy about 1%
- Removing any other column reduced accuracy below 90% for all methods
- Were able to increase testing accuracy from 0.9342 to 0.9714 by adding 2 more layers and increasing the nodes and epochs
- Dropout worked better than L2 regularization for our data