Developing an Artificial Intelligence to Classify Emotion Across Text, Audio, and Video

Sameer Saxena
April 18, 2018
Sponsor: Kelley Bethoney, PhD.
Clare Foundation
The Episcopal Academy
1785 Bishop White Drive, Newtown Square, PA 19073

The Importance of Emotional Analytics

- Expression matters just as much as the content
- Abstract concept to recognize and understand
    - Humans have a hard time understanding themselves
- Emotion heavily influences our lives
    - Provides information
    - Results and triggers behavior
    - Guides interactions
    - Prompts decision-making
    - Leads to motivation or depression
- Professor Albert Mehrabian's Communication Theory:
    - 7% of meaning content-based
    - 38% of meaning paralinguistic
    - 55% of meaning facial expressions

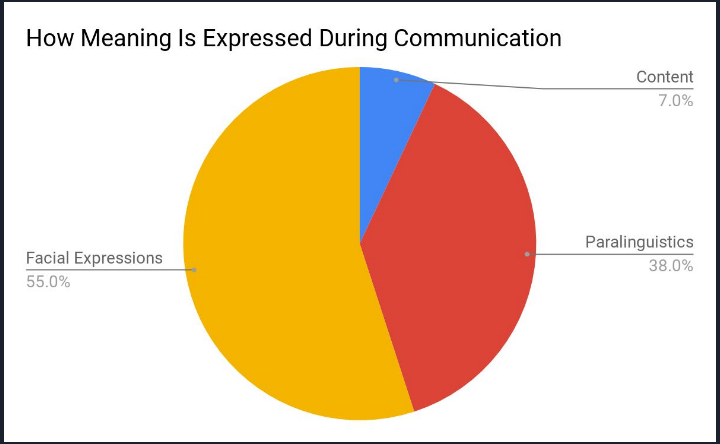


Figure 1: Pie chart to illustrate professor Albert Mehrabian's Communication Theory, which argues that meaning is expressed through far more than simply content of speech.

Emotional Analysis for Computers

- Growth of Artificial Intelligence
  - Computers understand more complex/abstract concepts
  - More open-source
- Emotional Analysis: Computer understands how you **feel**
- Classification of your current sentiment
  - Neutral
  - Positive: Happiness, Surprise
  - Negative: Sadness, Fear
- Fields of CS involved: NLP, image analysis, sound processing, machine learning, artificial intelligence
- Opportunity for human-machine interfaces
- Increase of human-machine interactions

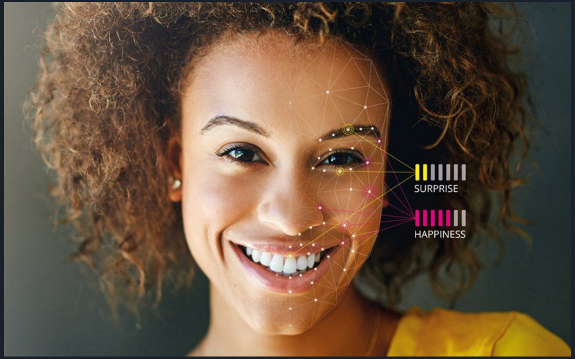


Figure 2: Illustrates example of emotion recognition based on facial expressions, done through artificial intelligence and image processing. Courtesy of Google Images

The Relevance of Emotional Analysis

- **Overall Goal:** To determine how a person reacts to the topic in which they are involved
- Businesses need to gauge a consumer response
  - Want to know how their product made the consumer feel
  - Market research on demographics which yield positive and negative responses
- Movies, books, TV shows, look through consumer reviews
  - Looking for positive and negative feedback
  - Judged by their reviews
- Politicians are able to measure their success through sentiment of public opinion
  - Overwhelmingly positive response = Successful campaign
- Celebrities understand how their actions influence the populace
  - Understand the positive and negative responses to their actions

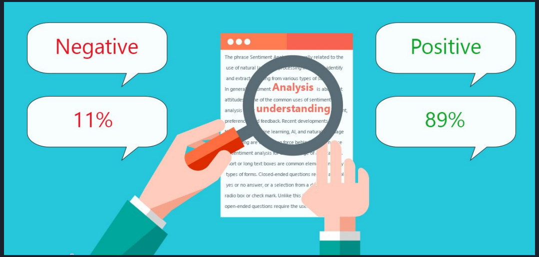


Figure 3: Graphic illustrating example of sentiment analysis on text (NLP). This example represents how businesses and politicians investigate for positive and negative responses in text. Courtesy of Google Images

Goal: To develop my own classifiers to perform emotional analysis on audio, video, and text.

The Basics of Machine Learning

- Teaching a computer to make predictions with data
- **Supervised Learning:** providing both a train set and test set for the computer to learn by matching the train set and applying itself through the test set
    - Train-set: Provide a set of data and labels to the computer
        - Computer learns how to match the data to the labels
    - Test-set: Provide a set of data to the computer without the labels
        - Computer attempts to predict the labels of the data given
    - Accuracy: Amount of correct labels / Total amount of labels for test-set
- Datasets are split into train-set and test-set
- Generalization: Computer must be as general as possible in order to be applied to data outside the scope of the dataset
- Computer develops a formula to match data to labels in the train-set
    - The **model** is the computer which executes this formula to predict labels of new data
    - The **classifier** is the method of deriving this formula

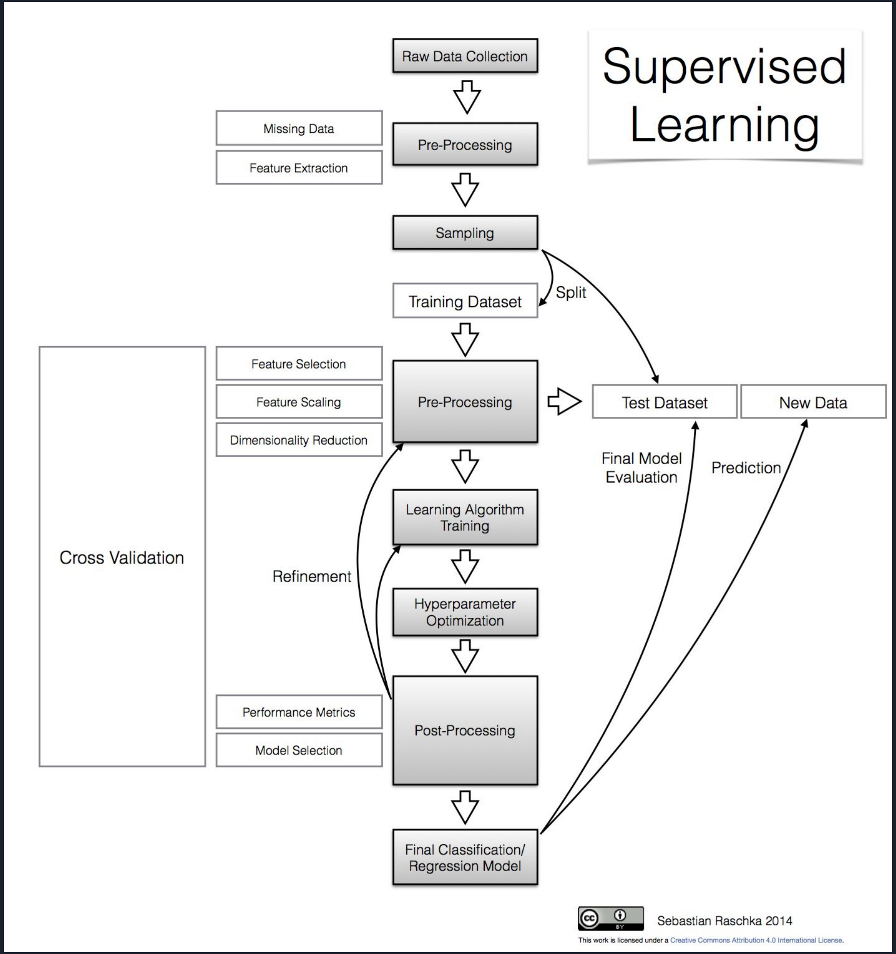
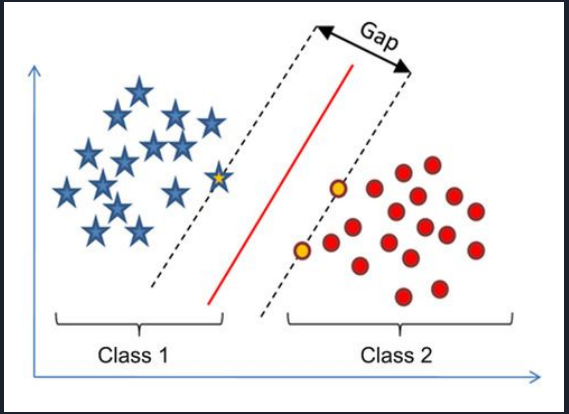


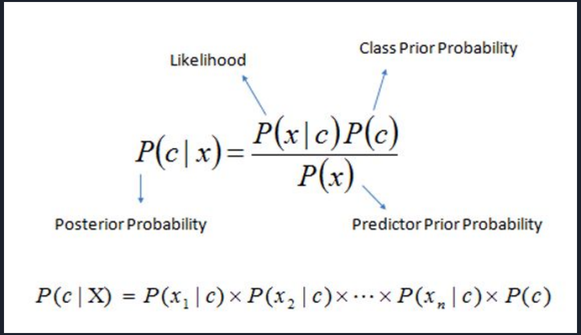
Figure 4: Flow chart illustrating the general process of supervised learning. Supervised learning is the most common type of machine learning. Courtesy of Google Images

Steps to Create Machine Learning Model

- Determine objective / goal of project
- Find dataset: large, reliable, maneuverable
- Iterate through data and process
  - Raw data can not be fed through a model
  - Text must be broken down into bag of words, images must be of certain dimensions, etc.
  - Results in multi-dimensional np (numpy) array
- Split processed data into train-set and test-set
  - For generalization, execute this process randomly
- Instantiate classifier to fit data on
- Feature selection: classifier takes the inputs of the training data and weighs them
- Hyperparameter tuning: classifier continues adjusting weights and other parameters to create most accurate model
- Test classifier on test-set, return accuracy

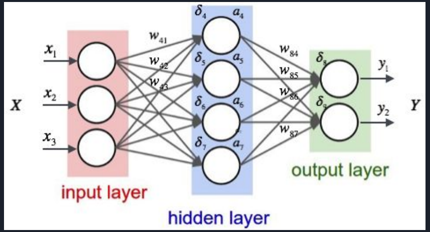
Different Types of Classifiers Can Perform This Task

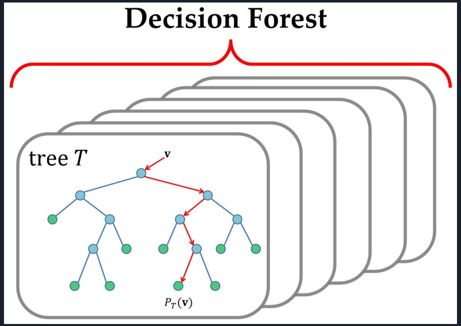




$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$





Figures 5-8: Each illustrates a different type of machine learning classifier. In the top left, there is an SVM, and in the top right, a Naive Bayes model. Both were utilized in this project. Underneath those two are (left to right) a single-layer neural network and a decision forest model. Courtesy of Google Images

My Approach To Complete the Project

- Create three models to classify each of the three forms of media
- Extract each form of media from input data
  - Input Data: Video
  - Extract video into transcript, audio, list of images
- Process all of the data to prepare it for classification
- Run data through ML models - retrieve prediction and confidence
- Weight each input in final model based on confidence and reliability
  - E.g. Audio may have more weight on final output if it has higher confidence

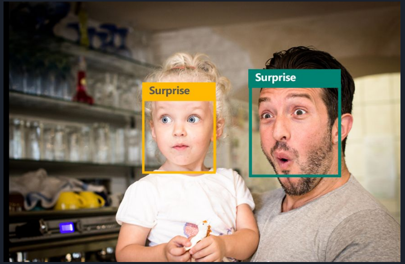


Figure 9: Example of emotional recognition based on facial expressions. Courtesy of Microsoft.

Flow Chart Illustrating Overall Product

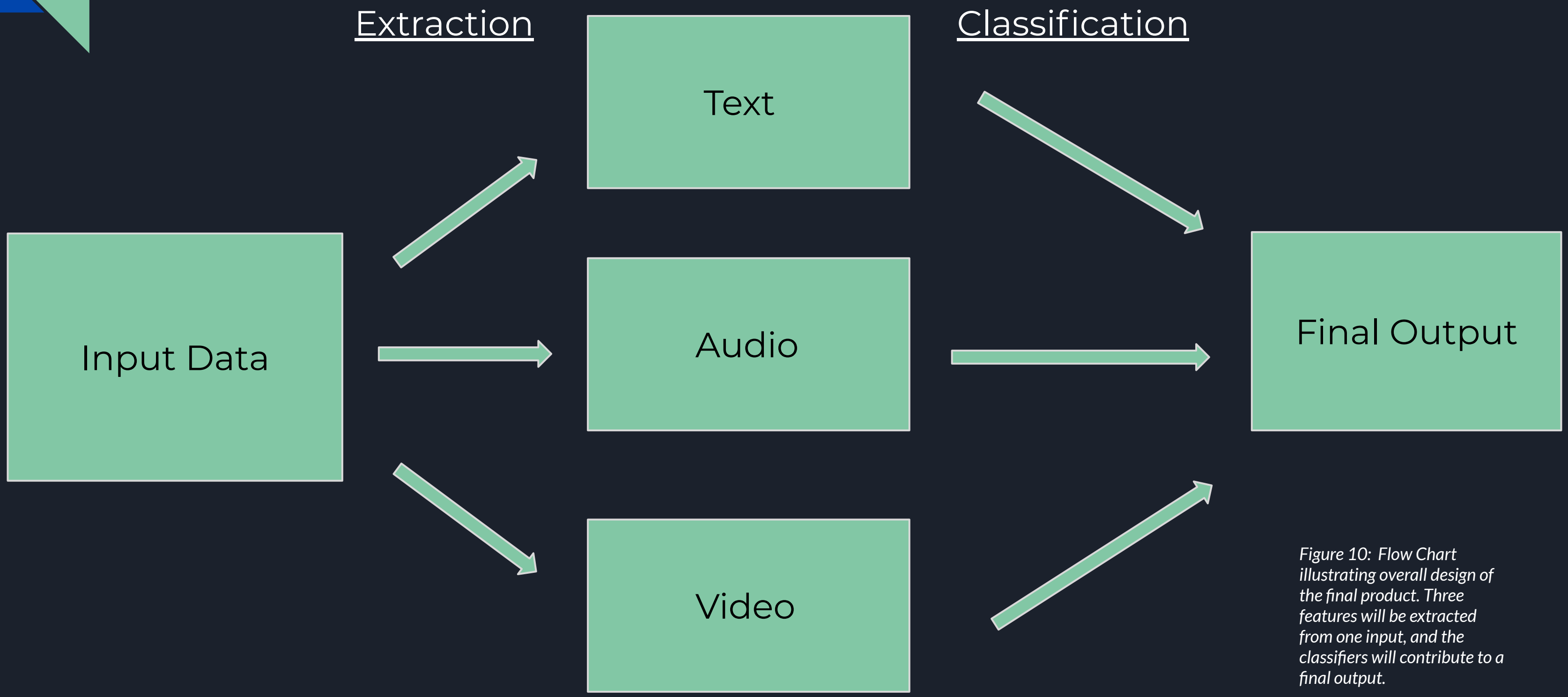


Figure 10: Flow Chart illustrating overall design of the final product. Three features will be extracted from one input, and the classifiers will contribute to a final output.

General Flow Chart For Classifiers

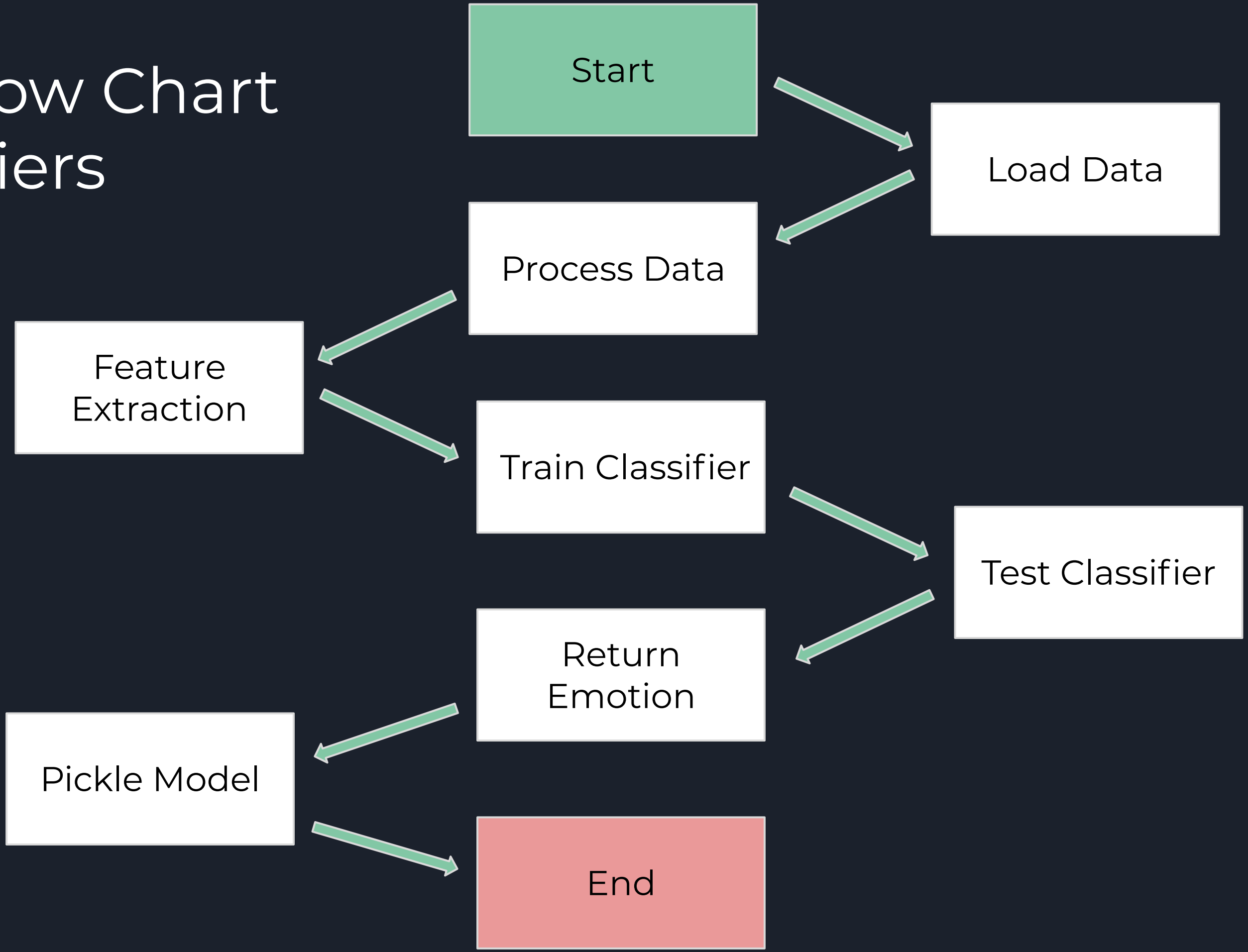


Figure 11: Flow Chart illustrating general design of each classification. This will apply to text classification, audio classification, and video classification. The only differences will be in the algorithms used.

Textual Sentiment Analysis Portion of Final Product

- **Big Picture:** From the input data, analyze the content of the speech as to determine whether the speaker is speaking positively or negatively
- **Objective:** Develop a model to classify statements as either positive or negative and provide its confidence in the decision
- Positive statements include…
  - "I am having a great day today."
  - "This movie was magnificent."
- Negative statements include…
  - "I am having an awful day today."
  - "This movie was terrible."
- **Insight: Sentiment heavily influenced by word choice**
  - "Great" and "magnificent" invoke positive sentiment
  - "Awful" and "terrible" invoke negative sentiment

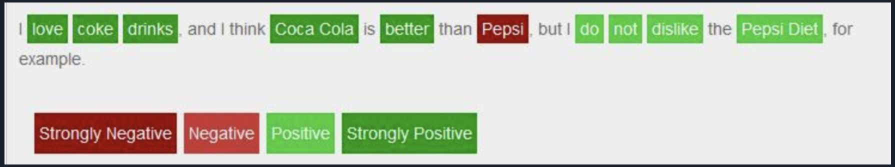

I love coke drinks, and I think Coca Cola is better than Pepsi, but I do not dislike the Pepsi Diet, for example.
Strongly Negative
Negative
Positive
Strongly Positive

Figure 12: Example of sentiment analysis breaking down sentences into words which can be scored on a negative to positive scale. Courtesy of Google Images

Datasets Utilized for Textual Sentiment Analysis

- Generalization - must choose dataset which could apply to real life conversations
- Twitter dataset - tagged list of positive and negative tweets from everyday life
  - **Pros:** Great generalization as it applied to real life scenarios
  - **Cons:** Bad grammar, incorrect spelling, etc. meant more time spent on data processing; Uneven balance between amount of positive and negative tweets (2x negative)
- Short Movie Reviews dataset - tagged list of short movie reviews
  - **Pros:** Obvious sentiment; keywords; large and balanced (5300 positive, 5300 negative)
  - **Cons:** Not great generalization as all of them were talking about movies

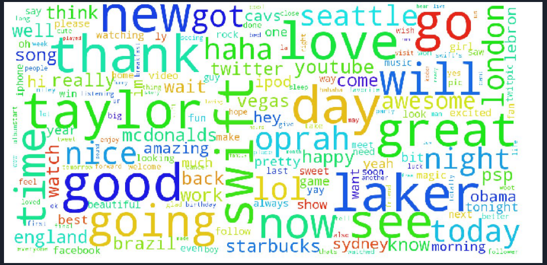


Figure 13: Generated WordCloud to illustrate the most frequently occurring words in the positive Twitter dataset.

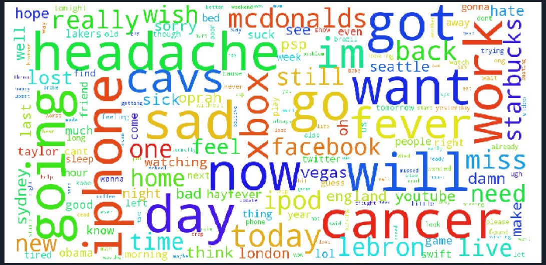


Figure 14: Generated WordCloud to illustrate the most frequently occurring words in the negative Twitter dataset.

Processing Data from Textual Datasets

- Remove @ symbols along with the handles after the @ symbol
- Expand contracts to make it easier for model to analyze keywords
  - E.g. “Can’t” = “Can not”
- Remove punctuation and numbers in sentences to prepare for tokenization
- Set all characters to lowercase
  - Model may differentiate Love and love as two different keywords
- Remove stop words which illustrate no emotion
  - E.g. “of” “that” “and”
- Filter sentence into only the most significant words that will show emotion
- Tokenize sentence through NLTK

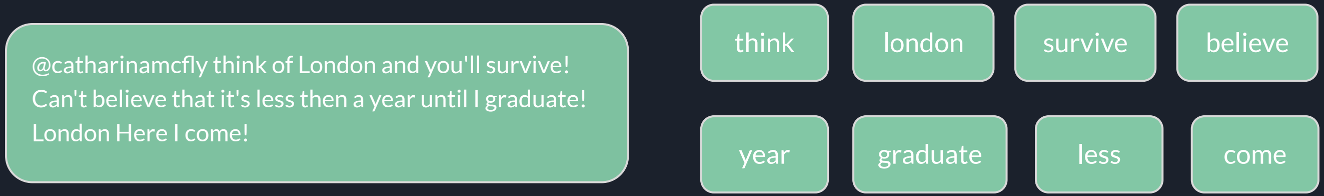

@catharinamcfly think of London and you'll survive! Can't believe that it's less then a year until I graduate! London Here I come!

Figure 15: Diagram to illustrate how tweets are processed and broken down to be passed into Naive Bayes Classifier. Input on left, outputs on right.

Methods of Data Processing on Textual Dataset

- **Word Frequencies** - iterated through positive and negative tweets, tokenized sentences into array of words, each word added to positive or negative list
- **Bigram Frequencies**
  - Bigrams were utilized in order to include “not good” = negative and “not wait” = positive
  - Bigrams were collected by taking all combinations of two words in the statement
  - Frequencies of bigrams taken and only top 10,000 bigrams accounted for
- **Bag of (Best) Words/Bigrams model**
  - Scores each word and bigram through BigramAssocMeasures.chi_sq
  - Computer only takes into account top 10,000 scored words/bigrams when tokenizing sentences
- When given new statement - computer uses scores of each “best” word/bigram to determine positive or negative overall

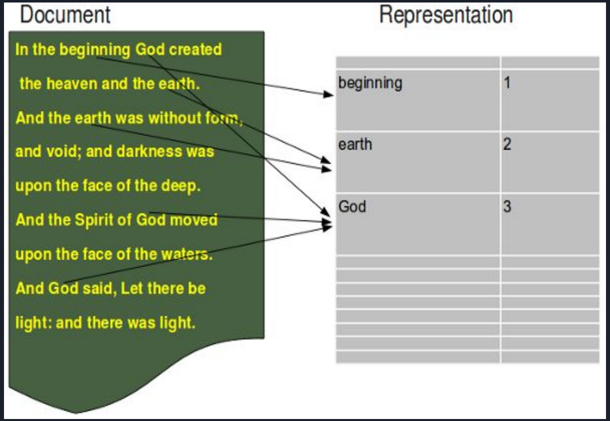
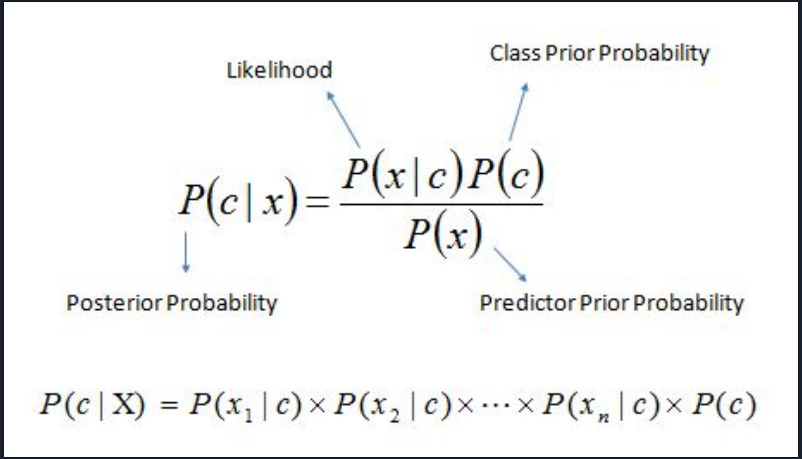

Document
Representation
In the beginning God created
the heaven and the earth.
And the earth was without form,
and void; and darkness was
upon the face of the deep.
And the Spirit of God moved
upon the face of the waters.
And God said, Let there be
light: and there was light.
beginning
1
earth
2
God
3

Figure 16: Diagram to represent bag of words model where sentences are broken down into their best words, which are then each scored by the classifier. Courtesy of Google Images

Creation of Naive Bayes Classifier Model for Textual Sentiment Analysis

- Randomly shuffled data into train_set and test_set
  - train_set : 75% of dataset
  - test_set : 25% of dataset
- Imported a **NaiveBayesClassifier** from NLTK

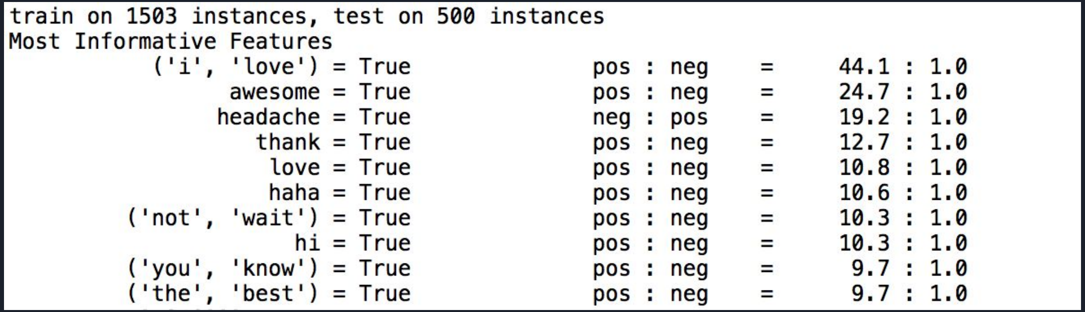


$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Figure 17: Diagram to illustrate formula behind Naive Bayes Classifier, a probabilistic model. Courtesy of Google Images

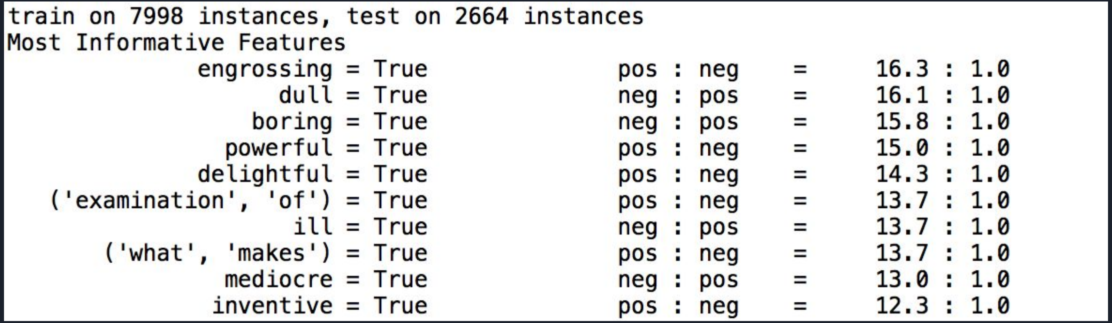
Most Informative Features of NB Classifier



```
train on 1503 instances, test on 500 instances
Most Informative Features
           ('i', 'love') = True              pos : neg    =     44.1 : 1.0
                 awesome = True              pos : neg    =     24.7 : 1.0
                headache = True              neg : pos    =     19.2 : 1.0
                   thank = True              pos : neg    =     12.7 : 1.0
                    love = True              pos : neg    =     10.8 : 1.0
                    haha = True              pos : neg    =     10.6 : 1.0
         ('not', 'wait') = True              pos : neg    =     10.3 : 1.0
                      hi = True              pos : neg    =     10.3 : 1.0
         ('you', 'know') = True              pos : neg    =      9.7 : 1.0
         ('the', 'best') = True              pos : neg    =      9.7 : 1.0
```

Most informative features of Twitter dataset

Figures 18A and 18B: Screenshots of output which conveyed the most informative features of the classifier on each dataset. These words contributed the highest scores in the classifier.



```
train on 7998 instances, test on 2664 instances
Most Informative Features
             engrossing = True              pos : neg    =     16.3 : 1.0
                   dull = True              neg : pos    =     16.1 : 1.0
                 boring = True              neg : pos    =     15.8 : 1.0
               powerful = True              pos : neg    =     15.0 : 1.0
             delightful = True              pos : neg    =     14.3 : 1.0
    ('examination', 'of') = True              pos : neg    =     13.7 : 1.0
                    ill = True              neg : pos    =     13.7 : 1.0
        ('what', 'makes') = True              pos : neg    =     13.7 : 1.0
               mediocre = True              neg : pos    =     13.0 : 1.0
              inventive = True              pos : neg    =     12.3 : 1.0
```

Most informative features of Short Movie Reviews dataset

Results of Naive Bayes Classifier on Each Dataset

- On the Twitter dataset - achieved a mean accuracy of 81% on validation set
- On the short movie reviews dataset - achieved a mean accuracy of 82% on validation set

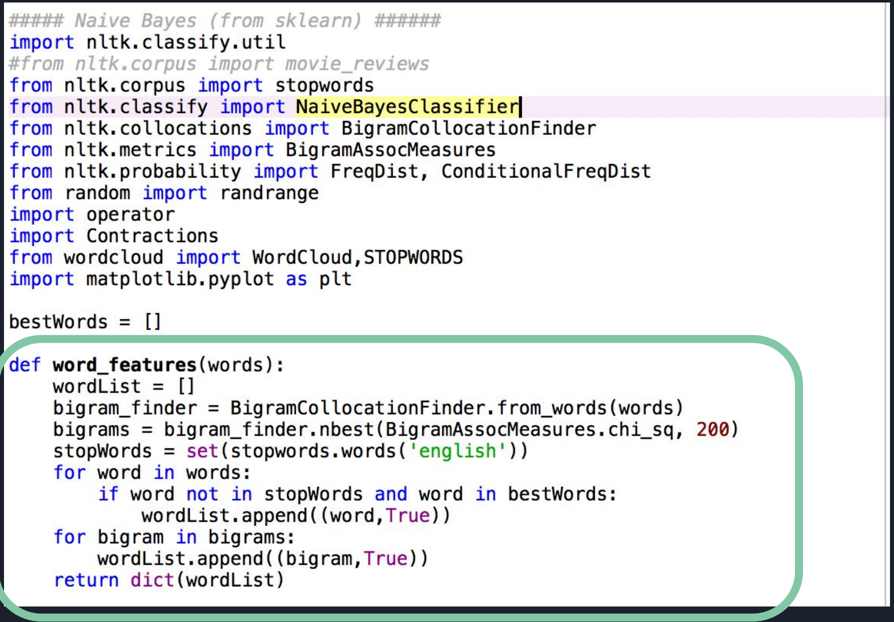
Predicted each statement as either positive or negative. Did not take into account neutral statements

Trial Number	Twitter Dataset	Movie Dataset
1	82.80%	82.02%
2	81.00%	81.61%
3	79.60%	82.24%
4	80.00%	82.30%
5	81.20%	83.00%

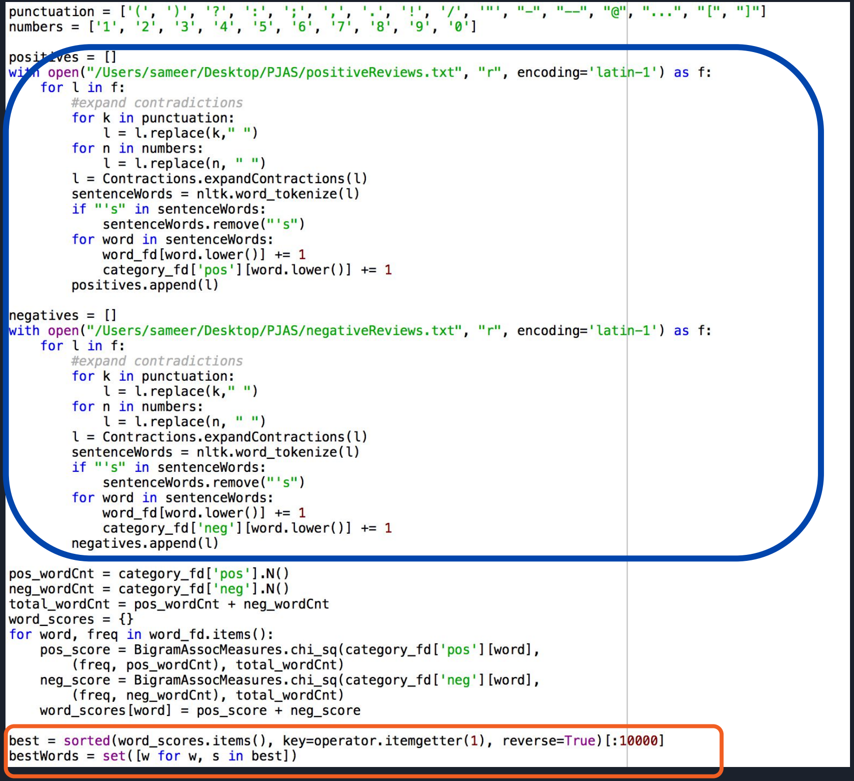
Figure 19: Data table of five trials run to determine mean accuracy of Naive Bayes Classifier on both Twitter and Short Movie Reviews dataset

Code for Loading Dataset and Processing Data

Figures 20 - 22: Screenshotted code for textual sentiment analysis

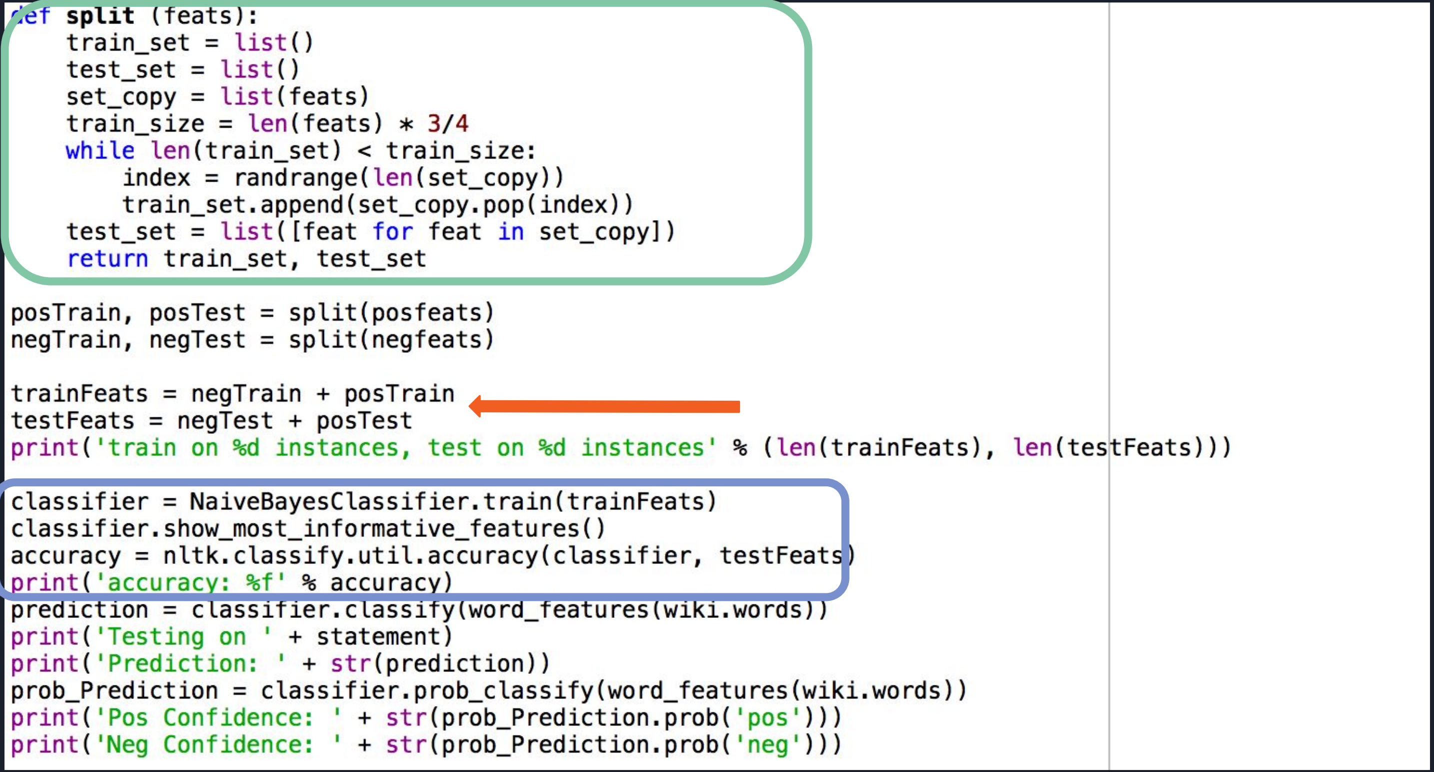


```python
##### Naive Bayes (from sklearn) ######
import nltk.classify.util
#from nltk.corpus import movie_reviews
from nltk.corpus import stopwords
from nltk.classify import NaiveBayesClassifier
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
from random import randrange
import operator
import Contractions
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt

bestWords = []

def word_features(words):
    wordList = []
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(BigramAssocMeasures.chi_sq, 200)
    stopWords = set(stopwords.words('english'))
    for word in words:
        if word not in stopWords and word in bestWords:
            wordList.append((word,True))
    for bigram in bigrams:
        wordList.append((bigram,True))
    return dict(wordList)
```



```python
punctuation = ['(', ')', '?', ':', ';', ',', '.', '!', '/', '"', "-", "--", "@", "...", "[", "]"]
numbers = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']

positives = []
with open("/Users/sameer/Desktop/PJAS/positiveReviews.txt", "r", encoding='latin-1') as f:
    for l in f:
        #expand contradictions
        for k in punctuation:
            l = l.replace(k," ")
        for n in numbers:
            l = l.replace(n, " ")
        l = Contractions.expandContractions(l)
        sentenceWords = nltk.word_tokenize(l)
        if "'s" in sentenceWords:
            sentenceWords.remove("'s")
        for word in sentenceWords:
            word_fd[word.lower()] += 1
            category_fd['pos'][word.lower()] += 1
        positives.append(l)

negatives = []
with open("/Users/sameer/Desktop/PJAS/negativeReviews.txt", "r", encoding='latin-1') as f:
    for l in f:
        #expand contradictions
        for k in punctuation:
            l = l.replace(k," ")
        for n in numbers:
            l = l.replace(n, " ")
        l = Contractions.expandContractions(l)
        sentenceWords = nltk.word_tokenize(l)
        if "'s" in sentenceWords:
            sentenceWords.remove("'s")
        for word in sentenceWords:
            word_fd[word.lower()] += 1
            category_fd['neg'][word.lower()] += 1
        negatives.append(l)

pos_wordCnt = category_fd['pos'].N()
neg_wordCnt = category_fd['neg'].N()
total_wordCnt = pos_wordCnt + neg_wordCnt
word_scores = {}
for word, freq in word_fd.items():
    pos_score = BigramAssocMeasures.chi_sq(category_fd['pos'][word],
        (freq, pos_wordCnt), total_wordCnt)
    neg_score = BigramAssocMeasures.chi_sq(category_fd['neg'][word],
        (freq, neg_wordCnt), total_wordCnt)
    word_scores[word] = pos_score + neg_score

best = sorted(word_scores.items(), key=operator.itemgetter(1), reverse=True)[:10000]
bestWords = set([w for w, s in best])
```

Programming the Naive Bayes Classifier



```
def split (feats):
    train_set = list()
    test_set = list()
    set_copy = list(feats)
    train_size = len(feats) * 3/4
    while len(train_set) < train_size:
        index = randrange(len(set_copy))
        train_set.append(set_copy.pop(index))
    test_set = list([feat for feat in set_copy])
    return train_set, test_set

posTrain, posTest = split(posfeats)
negTrain, negTest = split(negfeats)

trainFeats = negTrain + posTrain
testFeats = negTest + posTest
print('train on %d instances, test on %d instances' % (len(trainFeats), len(testFeats)))

classifier = NaiveBayesClassifier.train(trainFeats)
classifier.show_most_informative_features()
accuracy = nltk.classify.util.accuracy(classifier, testFeats)
print('accuracy: %f' % accuracy)
prediction = classifier.classify(word_features(wiki.words))
print('Testing on ' + statement)
print('Prediction: ' + str(prediction))
prob_Prediction = classifier.prob_classify(word_features(wiki.words))
print('Pos Confidence: ' + str(prob_Prediction.prob('pos')))
print('Neg Confidence: ' + str(prob_Prediction.prob('neg')))
```

81%

Mean accuracy of Naive Bayes Classifier for textual sentiment analysis

Auditive Emotion Classification to Interpret Speech Expression

- **Big Picture:** From the input data, extract audio to run through emotional classification
- **Objective:** Process each .wav file to classify sound as pertaining to certain emotion
- Allows us to investigate HOW the user spoke
- Emotions -
  - Neutral
  - Angry
  - Calm
  - Happy
  - Fearful
  - Sad
  - Disgusted
  - Surprised

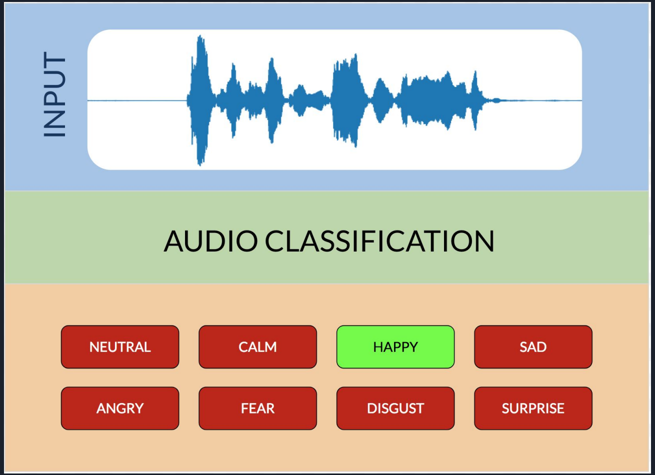
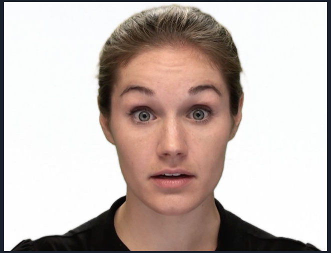
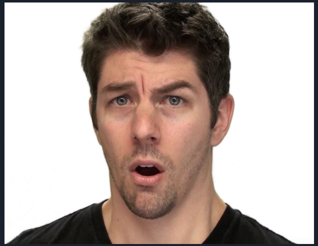


Figure 23: Self-made diagram to illustrate how audio classification took .wav file and classified it as one of eight possible emotions.

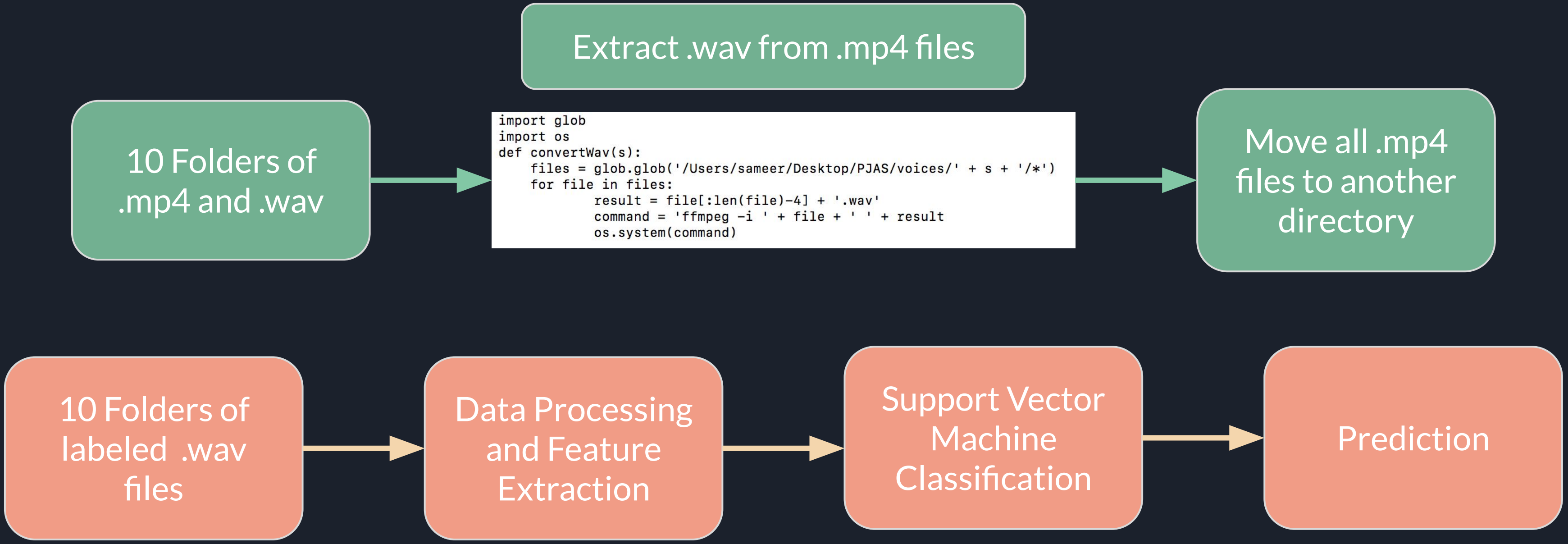
Data Processing on RAVDESS Dataset of Emotional Speech and Song

- Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)
- Fed over 3000 files of 10 different actors to model
- Actors both male and female
- Actors spoke each of the different emotions
- Split dataset into train_set (75%) and test_set (25%)
- Iterated through each file and extracted five features
  - MFCCS
  - Chromagram
  - Melspectrogram
  - Spectral Contrast
  - Tonal Centroid Features

Figures 23A - 23C: Still shots of .mp4 videos provided by RAVDESS dataset of actors speaking with different emotional expression.

Examining the Process of Audio Classification


Extract .wav from .mp4 files
10 Folders of .mp4 and .wav
Move all .mp4 files to another directory
10 Folders of labeled .wav files
Data Processing and Feature Extraction
Support Vector Machine Classification
Prediction

```
import glob
import os
def convertWav(s):
    files = glob.glob('/Users/sameer/Desktop/PJAS/voices/' + s + '/*')
    for file in files:
            result = file[:len(file)-4] + '.wav'
            command = 'ffmpeg -i ' + file + ' ' + result
            os.system(command)
```

Figure 24: Flow chart to illustrate the process of audio classification. Started from RAVDESS dataset and ended in creation of SVM classifier.

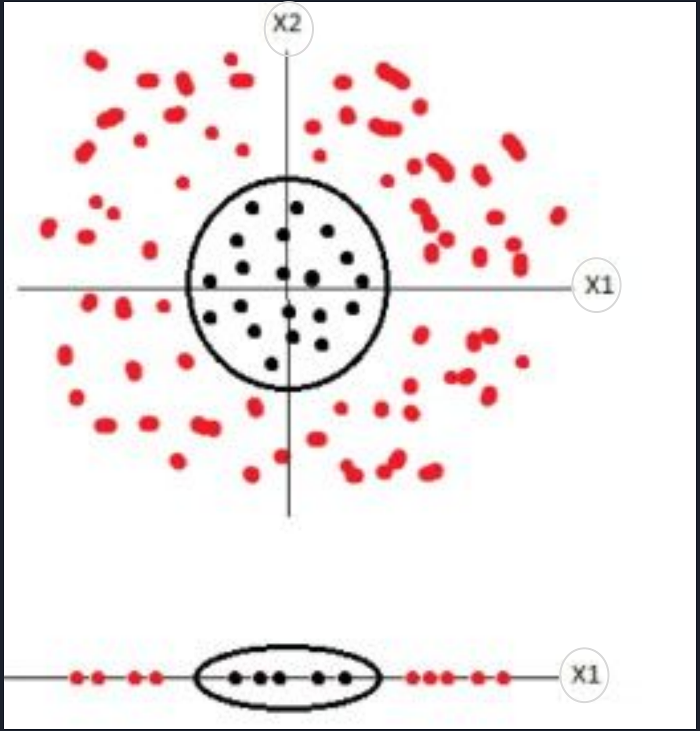
Implementing A Support Vector Machine to Classify Audio Files

- Fed data into SVM with a polynomial kernel
- Each .wav file plotted on a multi-dimensional graph (193)
- SVM creates a multi-dimensional kernel to separate classified groups

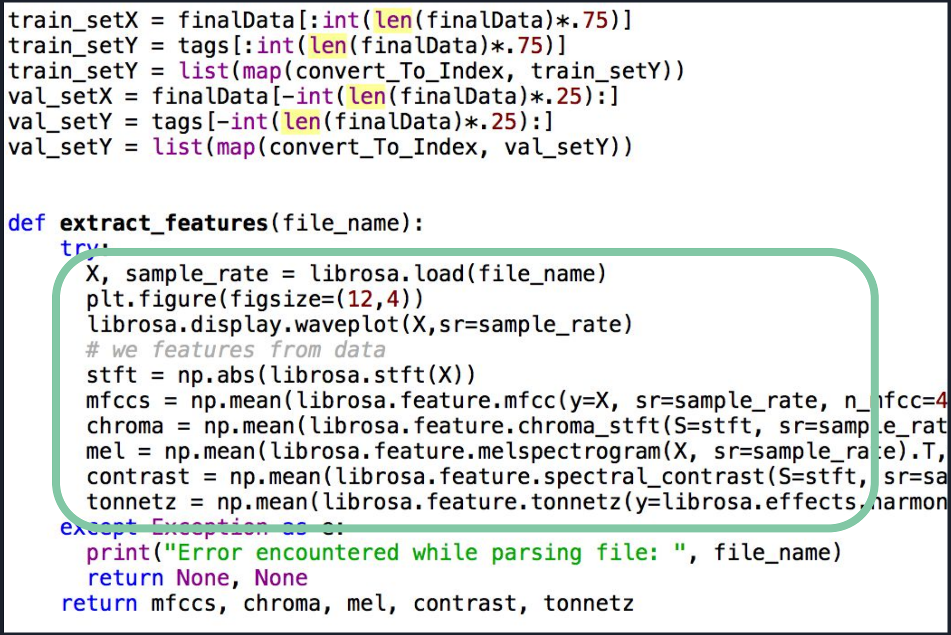
Trial Number	Accuracy Score
1	89.6%
2	91.0%
3	89.8%
4	89.2%
5	88.0%

Figure 25: Data table to show five trials to determine mean accuracy of SVM classifier.

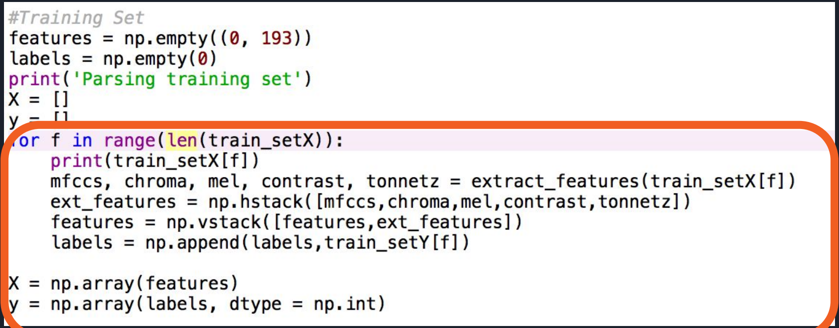
Figure 26: Illustration of SVM classifier in both 2D and 1D. Courtesy of Google Images



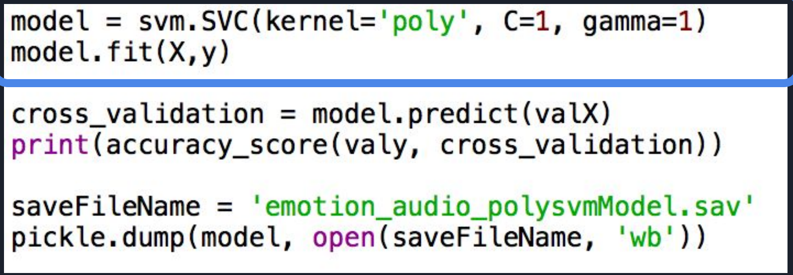
Writing Code for Feature Extraction and SVM Model Creation



```
train_setX = finalData[:int(len(finalData)*.75)]
train_setY = tags[:int(len(finalData)*.75)]
train_setY = list(map(convert_To_Index, train_setY))
val_setX = finalData[-int(len(finalData)*.25):]
val_setY = tags[-int(len(finalData)*.25):]
val_setY = list(map(convert_To_Index, val_setY))


def extract_features(file_name):
    try:
      X, sample_rate = librosa.load(file_name)
      plt.figure(figsize=(12,4))
      librosa.display.waveplot(X,sr=sample_rate)
      # we features from data
      stft = np.abs(librosa.stft(X))
      mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=4
      chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rat
      mel = np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,
      contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sa
      tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmon
    except Exception as e:
      print("Error encountered while parsing file: ", file_name)
      return None, None
    return mfccs, chroma, mel, contrast, tonnetz
```



```
#Training Set
features = np.empty((0, 193))
labels = np.empty(0)
print('Parsing training set')
X = []
y = []
for f in range(len(train_setX)):
    print(train_setX[f])
    mfccs, chroma, mel, contrast, tonnetz = extract_features(train_setX[f])
    ext_features = np.hstack([mfccs,chroma,mel,contrast,tonnetz])
    features = np.vstack([features,ext_features])
    labels = np.append(labels,train_setY[f])

X = np.array(features)
y = np.array(labels, dtype = np.int)
```



```
model = svm.SVC(kernel='poly', C=1, gamma=1)
model.fit(X,y)

cross_validation = model.predict(valX)
print(accuracy_score(valy, cross_validation))

saveFileName = 'emotion_audio_polysvmModel.sav'
pickle.dump(model, open(saveFileName, 'wb'))
```

Figures 26 - 28: Screenshotted code for audio emotional analysis

90%

Mean accuracy of Support Vector Machine for audio emotional analysis

Video Emotional Analysis to Classify Facial Expressions

- **Big Picture:** From the input data, cut video into images which can be run through facial recognition model
- **Objective:** Process each .png file to classify facial image as an expression of certain emotion
- Utilize the facial features of the speaker to classify emotion
- Emotions:
  - Neutral
  - Anger
  - Contempt
  - Disgust
  - Fear
  - Happiness
  - Sadness
  - Surprise

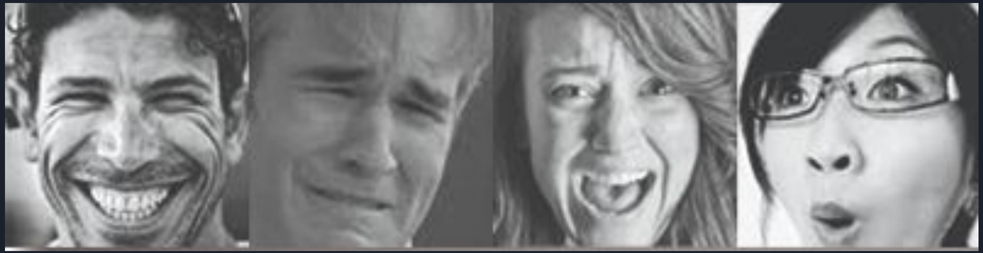


Figure 29: Examples of faces illustrating emotional expressions. Each of these images are processed and could be run through a classifier. Courtesy of Google Images

Datasets and Data Processing of Facial Images

- Downloaded Extended Cohn-Kanade Database which made small portion publicly available for developers
- Contained 593 sequences of 123 subjects
  - Sequences were the subjects' images going from neutral expression to expression of certain emotion
- Each sequence labeled by certain emotion
- Train_set (75%) and validation_set (25%)
- Processing of file:
  - Script to organize files into sets grouped by emotion
  - Iterated through images and detected face through HAARCascade
  - Cut image to only include face and converted to grayscale
  - Sent images through FisherFaceRecognizer model

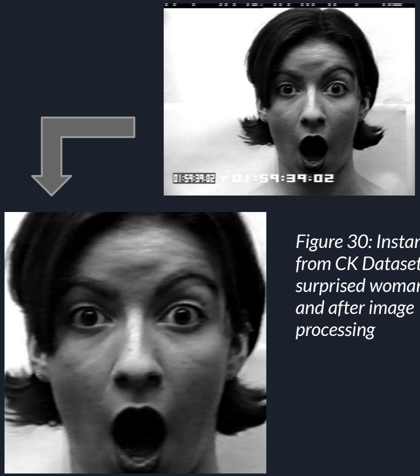


Figure 30: Instance from CK Dataset of surprised woman before and after image processing

Sorting, Processing, and Classifying the Facial Expression Dataset

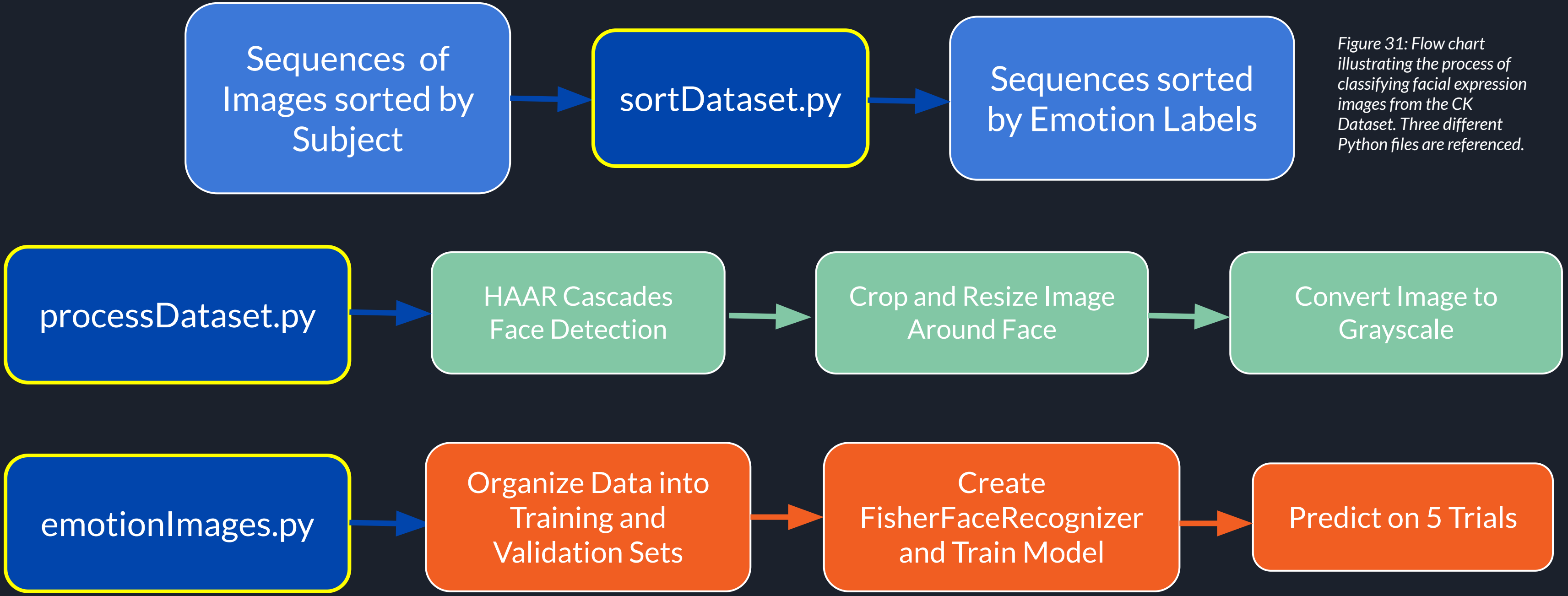

Sequences of Images sorted by Subject
sortDataset.py
Sequences sorted by Emotion Labels
processDataset.py
HAAR Cascades Face Detection
Crop and Resize Image Around Face
Convert Image to Grayscale
emotionImages.py
Organize Data into Training and Validation Sets
Create FisherFaceRecognizer and Train Model
Predict on 5 Trials

Figure 31: Flow chart illustrating the process of classifying facial expression images from the CK Dataset. Three different Python files are referenced.

Fisher Face Recognizer to Analyze Facial Expressions

- Convert each image to a fisher face which can be identified by computer
- Fisher faces are useful to exclude many variations in the image (want images to be as general and uniform as possible)
  - Variation in lighting
  - Variation in facial features
  - Angle between face and camera
  - Facial hair and hairstyles
- Fisher face recognizers separate classes linearly and therefore achieve higher accuracy

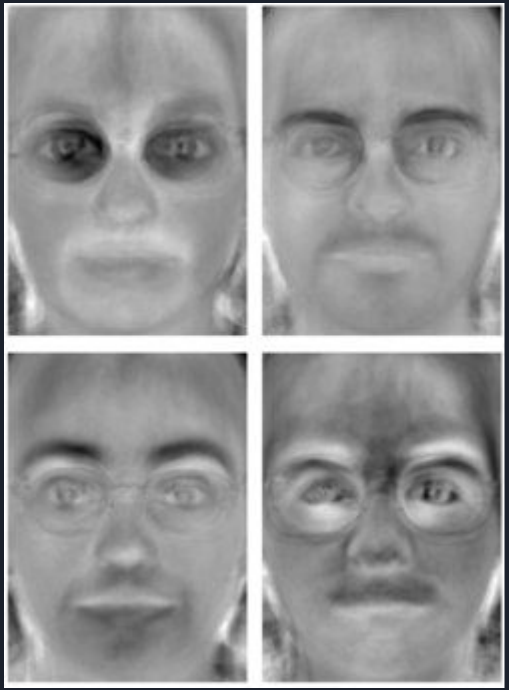


Figure 32: Example of the fisherface image processing to allow images to pass through FisherFaceRecognizer. Courtesy of Scholarpedia

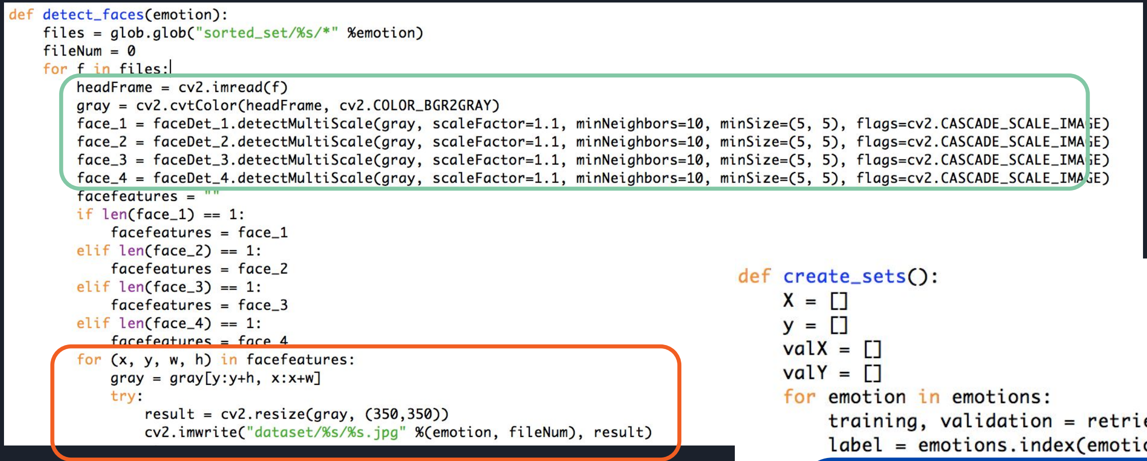
Results of Fisher Face Recognizer Over 10 Trials

- Ran Fish Face Recognizer 10 times over randomly shuffled dataset split into (75% and 25%)
- Iterated over validation set and calculated amount correct and amount incorrect
- Achieved accuracy consistently in between 78% to 83%

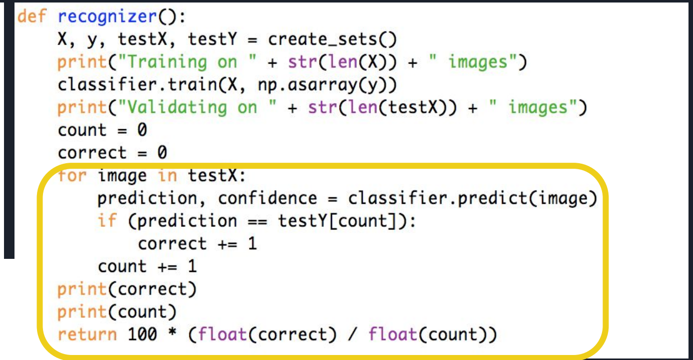
Trial Number	Accuracy
1	83.13%
2	83.75%
3	79.38%
4	80.00%
5	83.75%
6	78.13%
7	79.38%
8	83.75%
9	82.5%
10	79.38%

Figure 33: Data table to show ten trials of image classification and return mean accuracy.

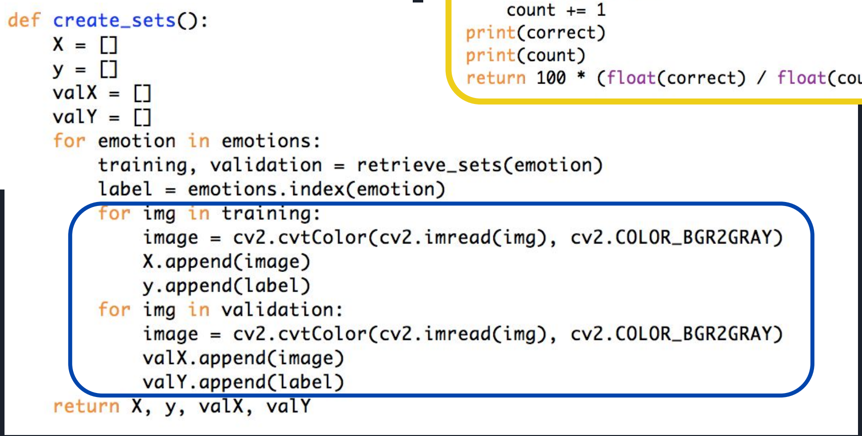
Programming for Facial Detection and Creation of Fisher Face Recognizer



```python
def detect_faces(emotion):
    files = glob.glob("sorted_set/%s/*" %emotion)
    fileNum = 0
    for f in files:
        headFrame = cv2.imread(f)
        gray = cv2.cvtColor(headFrame, cv2.COLOR_BGR2GRAY)
        face_1 = faceDet_1.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
        face_2 = faceDet_2.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
        face_3 = faceDet_3.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
        face_4 = faceDet_4.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=10, minSize=(5, 5), flags=cv2.CASCADE_SCALE_IMAGE)
        facefeatures = ""
        if len(face_1) == 1:
            facefeatures = face_1
        elif len(face_2) == 1:
            facefeatures = face_2
        elif len(face_3) == 1:
            facefeatures = face_3
        elif len(face_4) == 1:
            facefeatures = face_4
        for (x, y, w, h) in facefeatures:
            gray = gray[y:y+h, x:x+w]
            try:
                result = cv2.resize(gray, (350,350))
                cv2.imwrite("dataset/%s/%s.jpg" %(emotion, fileNum), result)
```



```python
def recognizer():
    X, y, testX, testY = create_sets()
    print("Training on " + str(len(X)) + " images")
    classifier.train(X, np.asarray(y))
    print("Validating on " + str(len(testX)) + " images")
    count = 0
    correct = 0
    for image in testX:
        prediction, confidence = classifier.predict(image)
        if (prediction == testY[count]):
            correct += 1
        count += 1
    print(correct)
    print(count)
    return 100 * (float(correct) / float(count))
```



```python
def create_sets():
    X = []
    y = []
    valX = []
    valY = []
    for emotion in emotions:
        training, validation = retrieve_sets(emotion)
        label = emotions.index(emotion)
        for img in training:
            image = cv2.cvtColor(cv2.imread(img), cv2.COLOR_BGR2GRAY)
            X.append(image)
            y.append(label)
        for img in validation:
            image = cv2.cvtColor(cv2.imread(img), cv2.COLOR_BGR2GRAY)
            valX.append(image)
            valY.append(label)
    return X, y, valX, valY
```

Figures 34-36: Screenshotted code of facial expression recognition

81%

Mean accuracy of Fisher Face Recognition on emotional analysis of video

Gathering Results to Develop Final Product

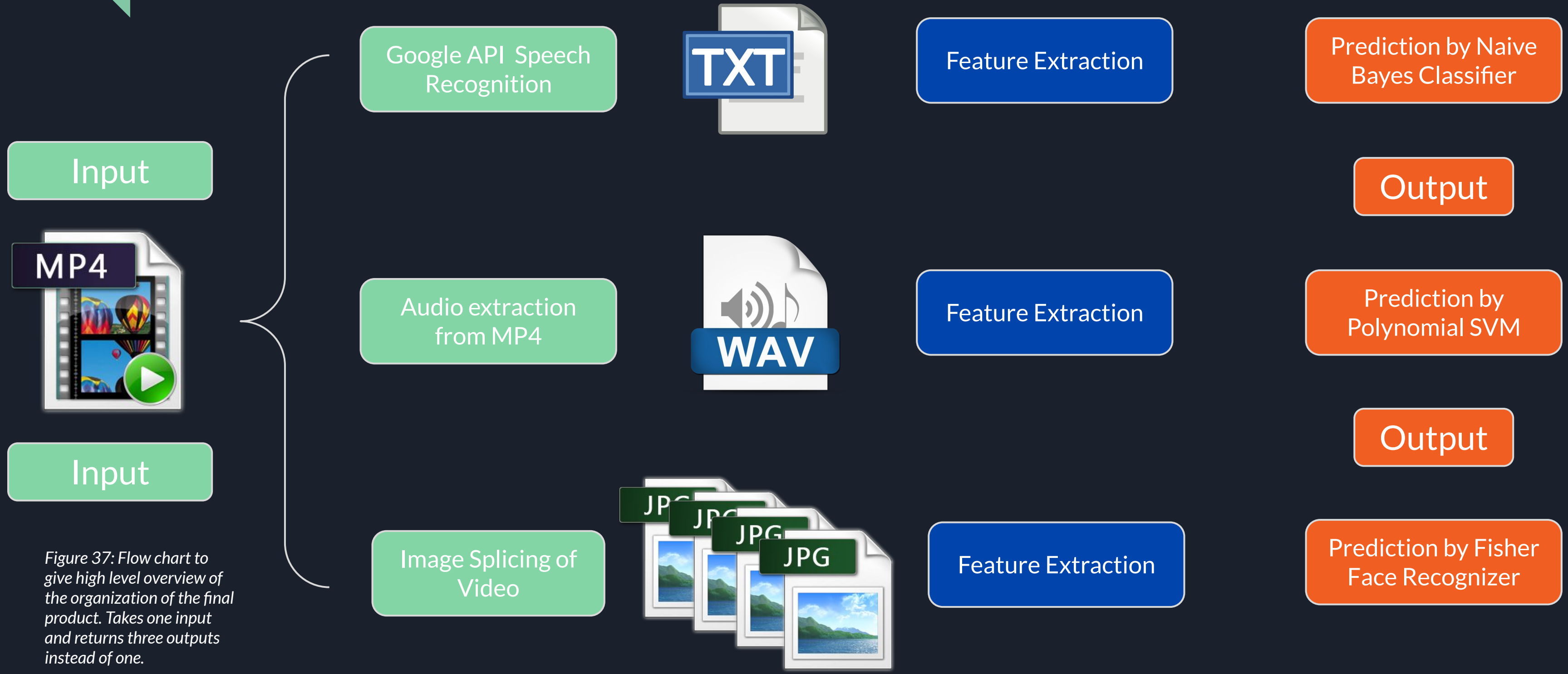

Input
MP4
Input
Google API Speech Recognition
Audio extraction from MP4
Image Splicing of Video
TXT
WAV
JPG
Feature Extraction
Feature Extraction
Feature Extraction
Prediction by Naive Bayes Classifier
Output
Prediction by Polynomial SVM
Output
Prediction by Fisher Face Recognizer

Figure 37: Flow chart to give high level overview of the organization of the final product. Takes one input and returns three outputs instead of one.

Application of Final Product to Real Life Scenarios

- Allows to gauge a response over all forms of media
    - Businesses: Analyze consumer responses
    - Entertainment Industry: Analyze viewer responses
    - Politicians: Analyze voter sentiment
    - Celebrities: Analyze popular response
- Great deal of potential and utility in market research
- Media encompasses all forms of communication: text, audio, video
    - Critics submit reviews in text format
    - Passionate creators post influential videos to broadcast their opinions
- Automatic analysis to gauge response of **anything**
- Makes life easier for organizations to perform sentiment analysis
    - Rather than using multiple services, they can just use one

Future Improvements

- Improve on each model, increase accuracy
  - Better data
  - Better processing
  - Better classifier
- Eliminate use of third-party speech recognizer in final product
- Experiment with ML model to adjust weights on text, audio, and video in final product
  - Final product should have one output rather than three
  - NN to determine how text, audio, and image classifications contribute to the one output
- Run trials to investigate whether product can recognize emotion better than humans
- Apply emotional analysis to broader range of emotions
- Apply product to investigate classification of sarcasm, deception, pain, etc.

Acknowledgements

PJAS Region 1A

PJAS Judges

PJAS Audience

Clare Foundation

Dr. Kelley Bethoney, PhD.

Mr. Matthew Memmo

Citations

- https://www.wired.com/insights/2013/09/emotions-analytics-to-transform-human-machine-interaction/
- https://www.businessballs.com/communication-skills/mehrabians-communication-theory-verbal-non-verbal-body-language-152/
- https://www.kdnuggets.com/2016/10/emotion-analytics-why-important.html
- https://www.socialmediatoday.com/content/who-benefits-sentiment-analysis
- https://www.kdnuggets.com/2016/05/voice-tone-analysis-emotion-detection.html
- https://www.analyticsvidhya.com/blog/2015/11/free-resources-beginners-deep-learning-neural-network/
- RAVDESS: https://dataverse.scholarsportal.info/dataset.xhtml?persistentId=doi:10.5683/SP/QJMWQ2
- https://www.kdnuggets.com/2016/09/urban-sound-classification-neural-networks-tensorflow.html
- https://soulhackerslabs.com/
- http://www.paulvangent.com/2016/04/01/emotion-recognition-with-python-opencv-and-a-face-dataset/
- CK: http://www.consortium.ri.cmu.edu/data/ck/CK/
- Twitter Dataset: https://www.kaggle.com/ngyptr/python-nltk-sentiment-analysis
- Short Movie Reviews: https://pythonprogramming.net/combine-classifier-algorithms-nltk-tutorial/?completed=/sklearn-scikit-learn-nltk-tutorial/