# Data Science for Humanities

## Classifying Depression

Alex Borgognoni
Daniel Gareev
Boghos Youseef

# Overview

- Introduction
  - Problem Statement
- Depression Classification
  - Dataset
  - Data Exploration
  - Data Processing / Feature Extraction
  - Fitting Models
    - Baseline Model
    - Gender Differences
    - Results
- Outlook

# Introduction

Depression recognition is an important topic of research as it helps bringing needed aid to those who need it at an early enough stage.

# Problem Statement

Can we **accurately predict** whether a person has **depression** based on the speech and non-speech variables?

# Project Objectives

- Classify whether a person is depressed based on the number of the speech and non-speech related features.
- See which variables of speech are the best predictors of depression symptoms.
- Explore a gender effect in speech variables and depression levels.

# Dataset

The dataset contains speech, non-speech features and clinical variables from participants of a depression study. Participants performed verbal tasks (recounting a positive, and negative event). The **_pos** and **_neg** tags indicate whether the feature is linked to the positive or negative event.

| | id int64 | age int64 | gender object | education int64 | dep_scale int64 | speech_ratio_neg float64 | speech_ratio_pos float64 | harmonics_to_noise_... |
|---|---|---|---|---|---|---|---|---|
| **0** | 2274 | 22 | female | 14 | 16 | 0.907168644 | 0.895367671 | 12.26756102 |
| **1** | 2275 | 21 | female | 13 | 15 | 0.883506084 | 0.87920034 | 14.36916863 |
| Expand rows 2 - 2 | | | | | | | | |
| **3** | 2282 | 23 | male | 16 | 12 | 0.842673164 | 0.836734694 | 4.73699753 |
| **4** | 2283 | 28 | male | 17 | 15 | 0.819487179 | 0.625386997 | 5.536165564 |

5 rows × 215 columns

The sample of the dataset

# Dataset

There are **121 participants** in the study and **215 features** in total in the dataset.

Each participant has an:

- ID
- Age
- Gender
- Education level
- Depression scale

| | id float64<br>121.0 − 3078.0 | age float64<br>6.513209717677321 − 121.0 | gender object<br>121 ........ 9.1%<br>3 others . 27.3%<br>Missing .... 63.6% | education float64<br>2.357801656233565 − 121.0 | dep_scale float64<br>5.6112465775681235 − 121.0 |
|---|---|---|---|---|---|
| **count** | 121.0 | 121.0 | 121 | 121.0 | 121.0 |
| **unique** | nan | nan | 2 | nan | nan |
| Expand rows 2 - 8 | | | | | |
| **75%** | 2980 | 26 | nan | 17 | 20 |
| **max** | 3078 | 75 | nan | 20 | 43 |

The initial description of the dataset

# Dataset

There are multiple different data points collected in this dataset. I will attempt to explain as many as possible

**id:** The unique ID number. values-range(2274, 3078)

**age:** The age in years. values-range(18, 75) mean = 24. average = 24.

**gender:** male or female. values-range(female, male)

**education:** the level of education (However, it is unclear how it is measured). values-range(8, 20)

**dep_scale:** values range from 6 to 43. values-range(6, 43)

# Dataset

### from here, every variable that ends with neg is a variable calculated while the person was recounting a negative experience
### while every variable that ends with pos is a variable calculated while the person was recounting a positive experience

**speech_ratio_neg:** the speech ratio of negative experiences being told. values-range(0.357697912, 0.961089494)

**speech_ratio_pos:** the speech ratio of positive experiences being told. values-range(0.572709163, 0.98374761)

**Harmonics to noise ratio:** "A Harmonicity object represents the degree of acoustic periodicity, also called Harmonics-to-Noise Ratio (HNR)"**[[1]]**
It can refer to both, the signal to noise ratio of any object that produces a periodic signal or, the quality of voice.

In the case of this data-set, it is the latter.

# Dataset

**harmonics_to_noise_ratio_neg:** quality of voice when recounting negative experiences. values-range(2.360100232, 14.69669859)

**harmonics_to_noise_ratio_pos:** quality of voice when recounting positive experiences. values-range(4.440765893, 15.55967063)

**Sound to Noise ratio**: is a measure that calculates the ratio of a desired signal to the background noise. It is usally measured in Decible. **[[2]]**

**sound_to_noise_ratio_neg:** values-range(-0.000562419, 0.000537104)

**[[2]]** Will Gragido, Johnl Pirc, Nick Selby, Daniel Molina,Chapter 4 - Signal-to-Noise Ratio.

# Dataset

**mean_f0_neg:**  values-range(68.92734293, 183.5693184)

**mean_f0_pos:**  values-range(71.56399653, 180.5547641)

**sd_f0_neg:**  values-range(5.208010699, 127.9854766)

**sd_f0_pos:**  values-range(6.391781829, 205.5167055)

# Dataset

*Total phonation time** "is a clinical measure of the longest time a person can phonate a vowel" **[[3]]**

**total_phonation_time_neg:** The total phonation time of individuals recounting negative experiences values-range(2.871369928, 370.2312181) measured in seconds

**total_phonation_time_pos:** The total phonation time of individuals recounting positive experiences values-range(3.865251701, 223.8618359) measured in seconds

**number_of_pauses_neg:**  values-range(0, 156) # self explanatory

**[[3]]** Jonathan Maslan, Xiaoyan Leng, Catherine Rees, David Blalock, Susan G. Butler,
Maximum Phonation Time in Healthy Older Adults

# Dataset

**number_of_pauses_pos:** values-range(0, 87)

**espinola_zero_crossing_metric_neg:** values-range(0, 17602)

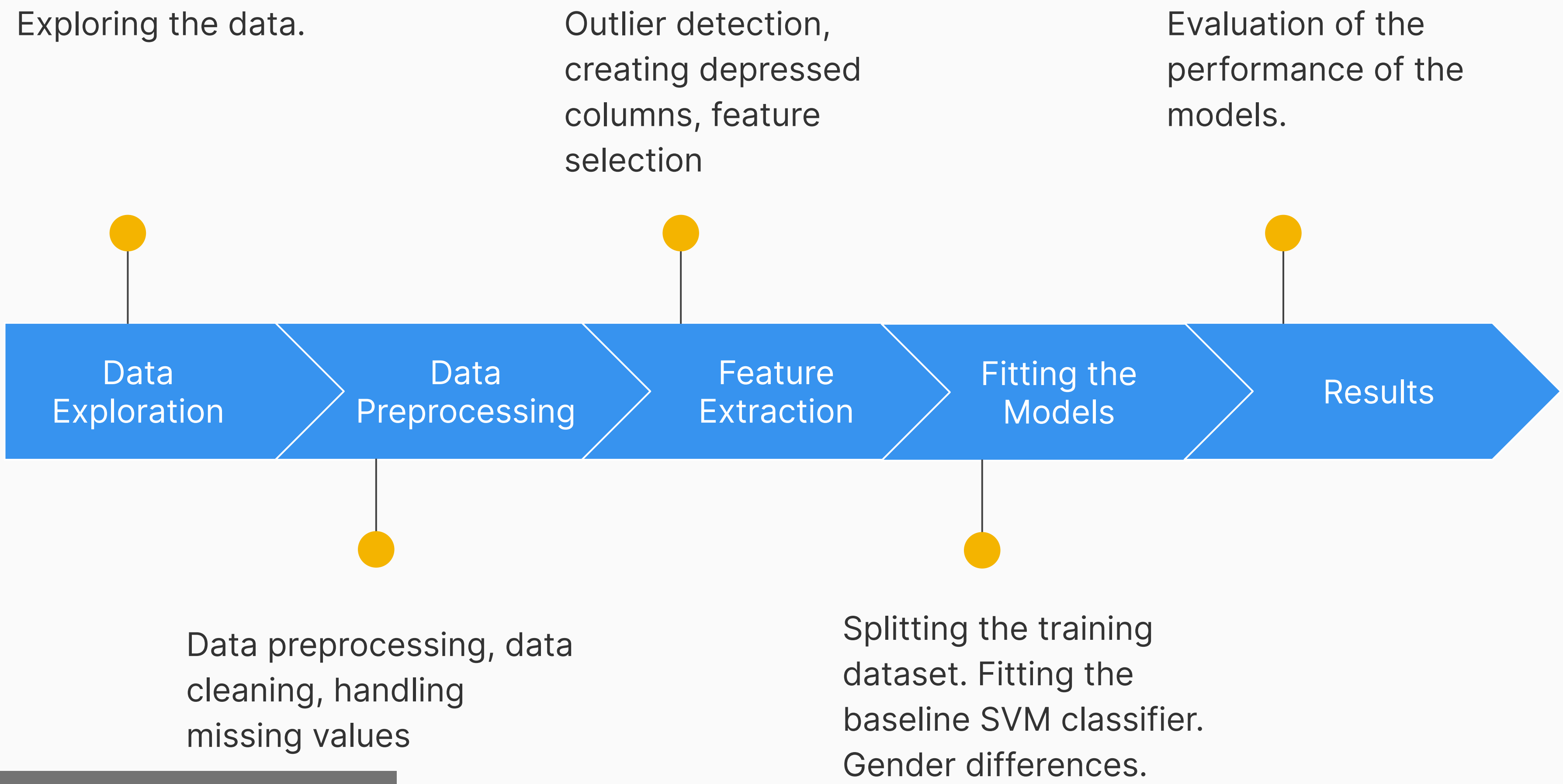
**espinola_zero_crossing_metric_pos:** values-range(0, 0)

**mfccs** is short for the Mel Frequency Cepstral Coefficient

"The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear." **[[4]]**

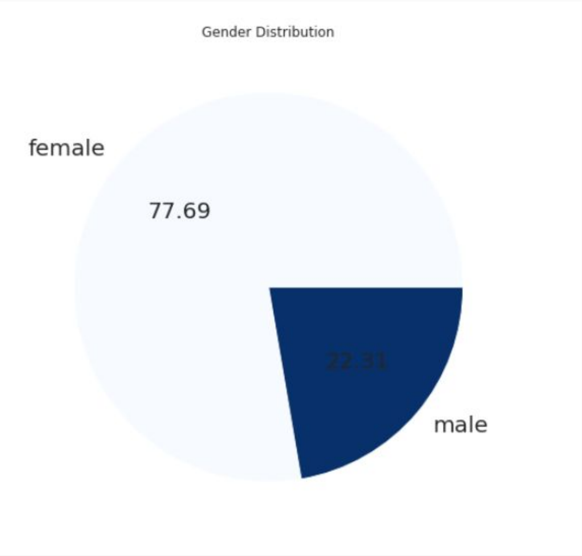
**[[4]]** http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

# Depression Classification



# Data Exploration

- Over 75% of the participants in the study are **females**.
- The average age of the participants is **24 years**.




Gender Distribution

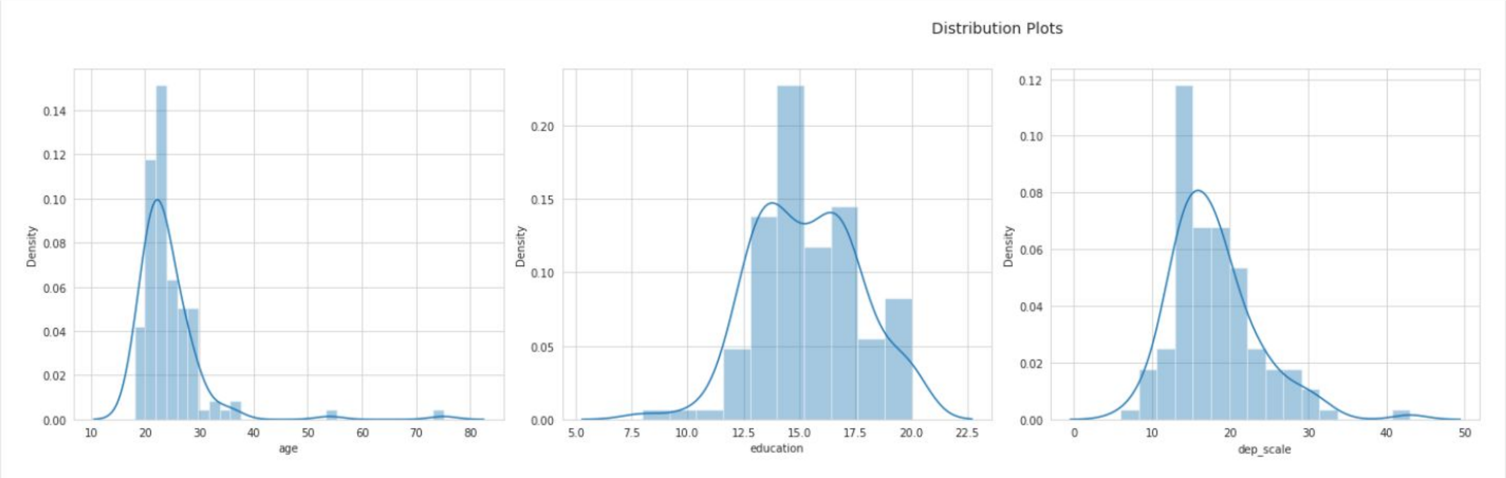
| | gender object | age float64 |
|---|---|---|
| 0 | female | 23.70212765957447 |
| 1 | male | 25.88888888888889 |

2 rows × 2 columns

Average age per gender

# Data Exploration

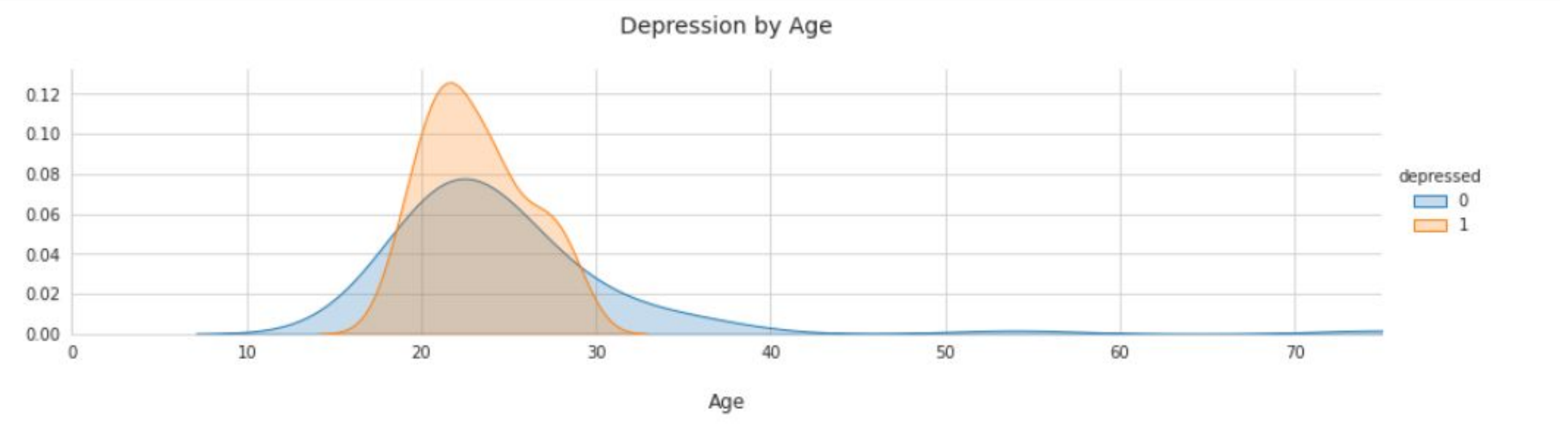
We checked the skewness of the variables by plotting the distribution plots. The dataset contains mostly adults in the age group between **20 to 40 years** old.




Distribution plots

# Data Exploration

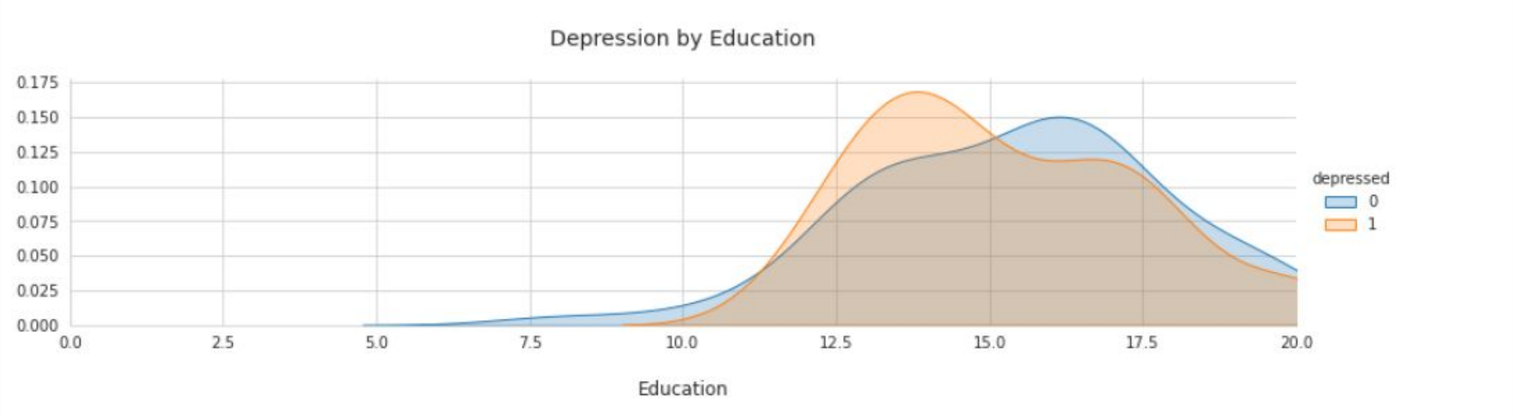
People in the age between 20 to 30 years are more likely to be depressed than any other group.




Depression by age

# Data Exploration

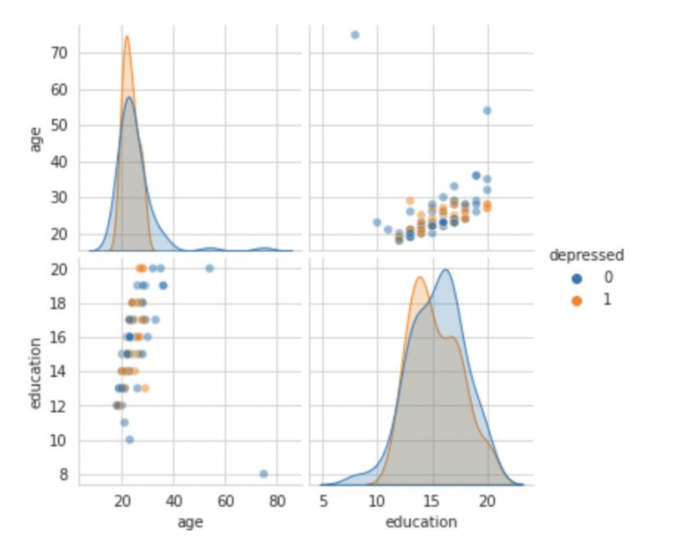
People are more likely to be depressed in the middle of the educational career. Moreover, the level of depression tends to decrease towards the end of the educational path.




Depression by age

# Data Exploration

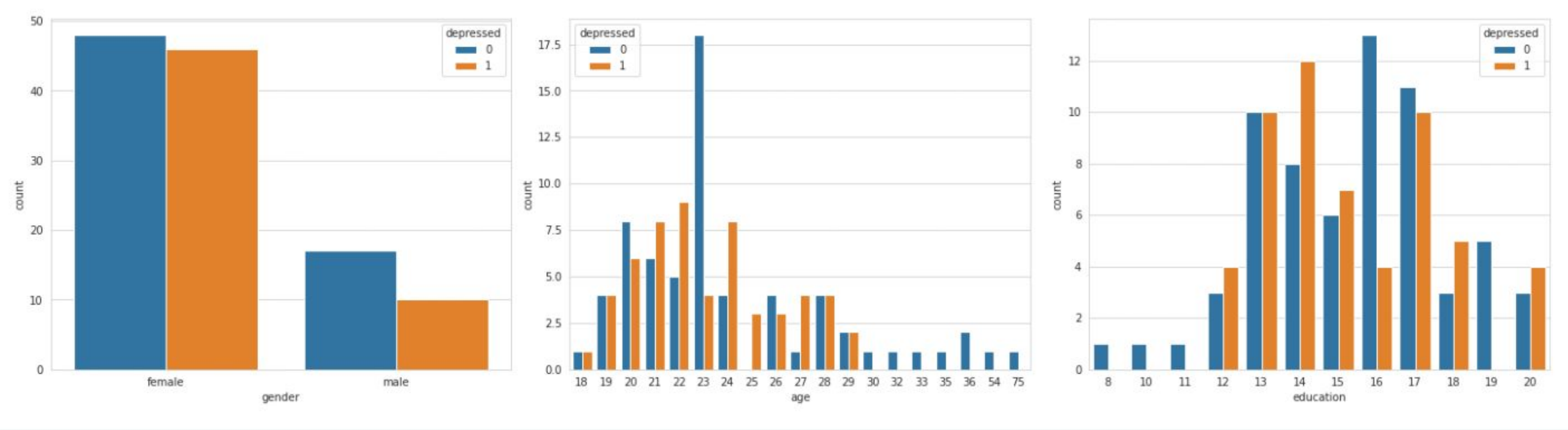
Here we can see that the older participants with higher education are less likely to get depressed.




Pairplot

# Data Exploration

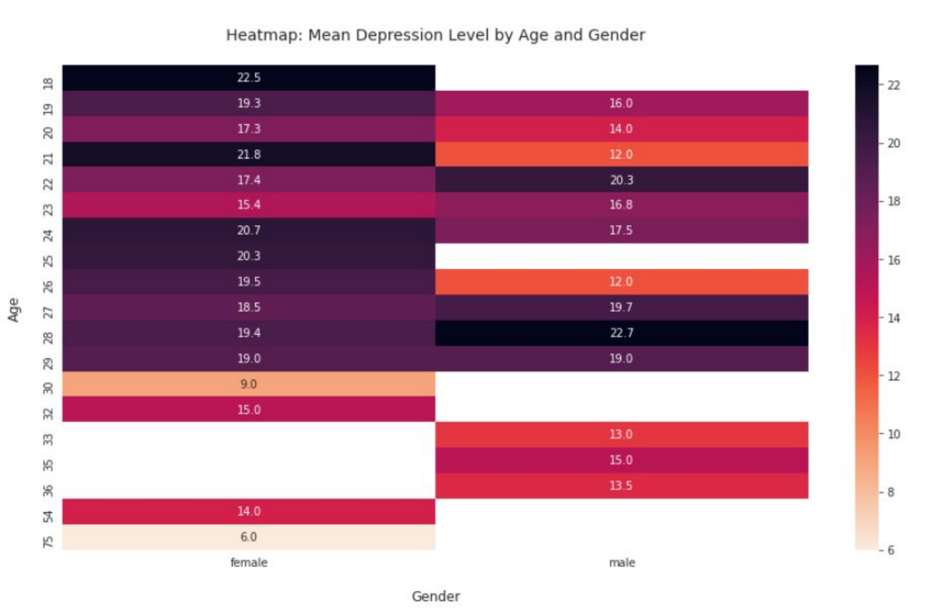
It appears that females are more depressed than males.




Pair plots

# Data Exploration

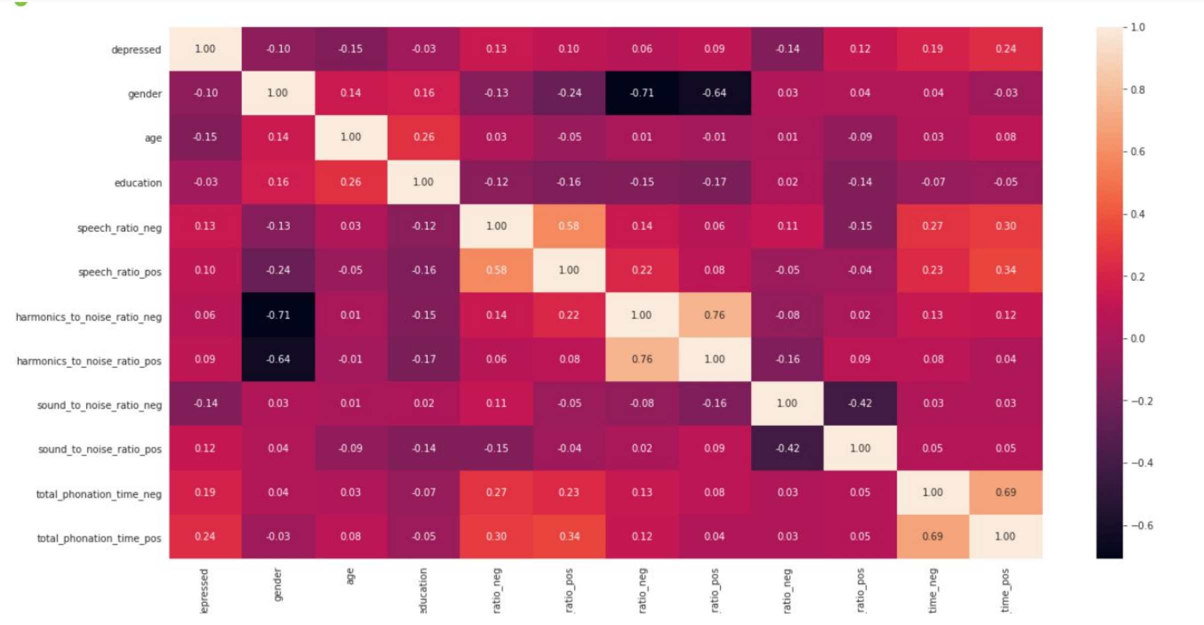
Female participants are the most depressed in the ages of 18 to 21, while male participants are the most depressed in the ages of 27-29.




Mean depression level by age and gender

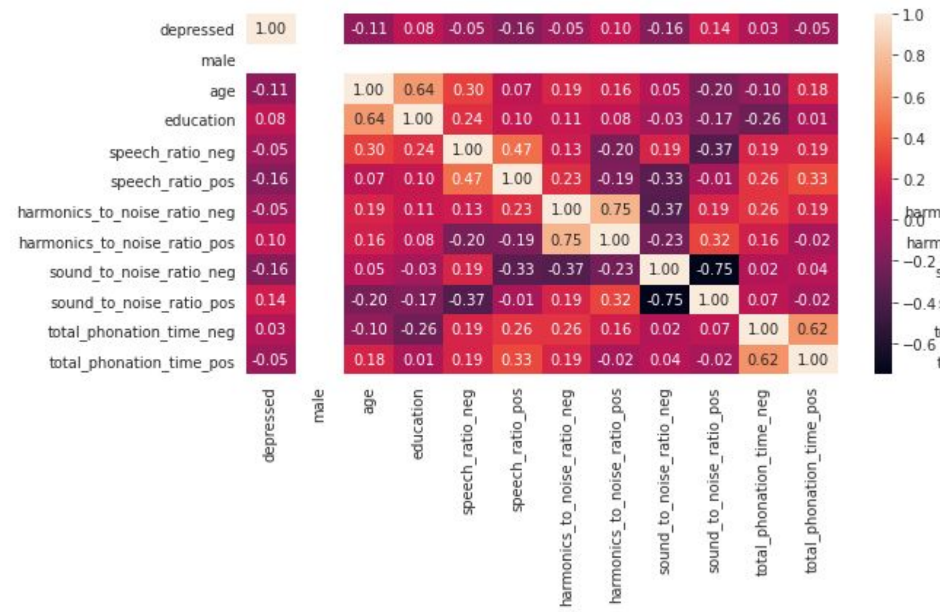
# Data Exploration

We can see that the depression is the most correlated with one of the speech variables, but not with the demographic features.

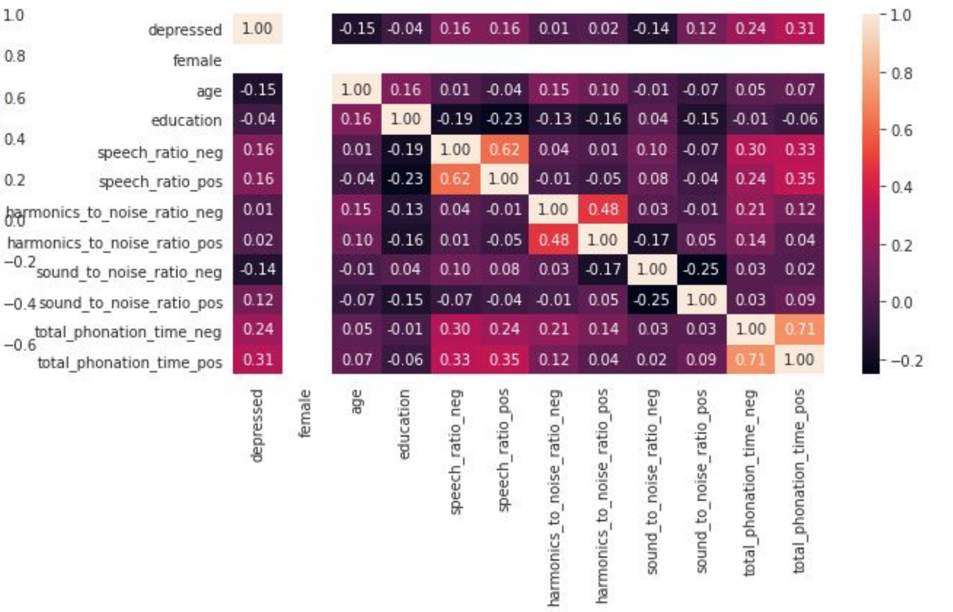



Feature correlation heatmap

# Data Exploration




Male feature correlation heatmap




Female feature correlation heatmap

# Data Processing

First, we converted the variable **gender** into binary, converted categorical variable **education** to one-hot encoding and dropped the **ID** of the participants as we don't need it for predicting.

```
# convert the categorical variable gender into binary variables.
genders = {'female': 0,'male': 1}
df.gender = [genders[item] for item in df.gender]

# use pandas get_dummies function to assign binary variables.
education_level = pd.get_dummies(df.education).astype(int)

# drop id and education columns
df = df.drop(['id', 'education'], axis = 1)

# drop the original columns and replace them with indicator columns
df = pd.concat((df, education_level), axis = 1)
```

Convert gender into binary, drop IDs

# Data Processing

Next, we created a column that would indicate whether a person is depressed or not based on the depression scale.

```
# create binary column 'depressed'
df['depressed'] = np.where(
    df['dep_scale'] <= 17, 0, np.where(
    df['dep_scale'] >  17, 1, None))
df["depressed"] = df["depressed"].astype(int)
```

Create binary column “depressed”

# Data Processing: Handling Missing Values

- **10 columns** with 10-12 missing values representing speech variables.
- The columns represent five kinds of jitter measurements, which are acoustic characteristics of voice signals.
- The missing values represent only **0.4%** of the dataset.

| | |
|---|---|
| jitter_local_neg | 10 |
| jitter_absolute_neg | 10 |
| jitter_rap_neg | 10 |
| jitter_ppq5_neg | 10 |
| jitter_ddp_neg | 10 |
| jitter_local_pos | 12 |
| jitter_absolute_pos | 12 |
| jitter_rap_pos | 12 |
| jitter_ppq5_pos | 12 |
| jitter_ddp_pos | 12 |

The number of NaN values in the columns

# Data Processing: Handling Missing Values

- The values are probably missing because the speech wasn't decoded for these features.
- We can guess the missing values based on the other values in that column and row rather than just leaving them as NA's.

| jitter_local_pos float64 | jitter_absolute_p… float… | jitter_rap_pos float64 | jitter_ppq5_pos float64 | jitter_ddp_pos float64 |
|---|---|---|---|---|
| nan | nan | nan | nan | nan |
| 0.055096318 | 0.000585023 | 0.028389441 | 0.026908859 | 0.085168324 |
| | | | | |
| nan | nan | nan | nan | nan |
| nan | nan | nan | nan | nan |

The sample of the rows with  NaN values.

# Data Processing: Handling Missing Values

- We used **Multiple Imputation** using MICE (Multiple Imputation by Chained Equations) to impute the missing data.
- Multiple imputation has a lot of advantages over traditional single imputation methods (e.g., mean, median).

```
# start the MICE training
imputed_training=mice(df.values)
```

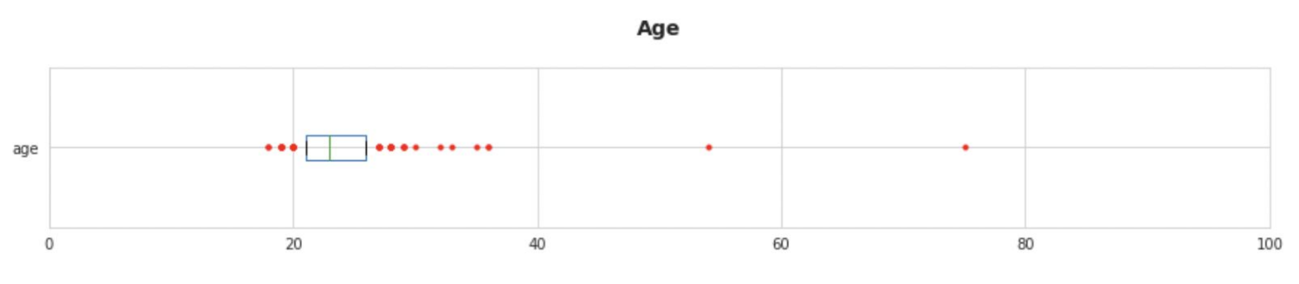
Imputing missing values

# Data Processing: Outlier Detection

- Started by exploring the demographic variables, such as age of the participants before moving to other features.
- The **75%** of the participants are up to **26 years old**, but the maximum age is **75**. We decided on a limit of 40 years old, and dropped all outliers that are older than that.

```
count    121.000000
mean      24.190083
std        6.513210
min       18.000000
25%       21.000000
50%       23.000000
75%       26.000000
max       75.000000
Name: age, dtype: float64
```

Description of “Age” column




Age distribution box plot

# Feature Extraction: Outlier Detection

- Next, we checked the correlations between our numerical features and see which features are highly correlated.
- We used a correlation matrix and convert the correlations to their absolute values in order to deal with negative correlations. We dropped the columns with the correlation of **95%** and above.

```
['number_of_pauses_neg',
 'number_of_pauses_pos',
 'mean_power_neg',
 'mean_power_pos',
 'total_power_neg',
 'total_power_pos',
 'jitter_ppq5_neg',
 'jitter_ddp_neg',
 'jitter_ppq5_pos',
 'jitter_ddp_pos',
 'avg_dependencies_neg',
 'avg_dependencies_pos',
 'mean_cluster_density_neg',
 'mean_cluster_density_pos',
 'biggest_cluster_density_neg',
 'biggest_cluster_density_pos',
```

Columns with a correlation higher than 95%

# Fitting the Models: Splitting the Data

Next, we splitted the dataset. The first dataset contains only speech features. The second dataset contains both speech and non-speech features (i.e., demographic and clinical).

```
[34]
# create target and features
demographic_clinical_feats = ['gender', 'dep_scale', 'age', 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
df_demographic_clinical_feats = pd.DataFrame(df, columns = demographic_clinical_feats)

# create dataframe with speech features
df_speech = df.drop(columns = demographic_clinical_feats)
df_speech = df_speech.drop(['depressed'], axis = 1)

# create dataframe with demographical, clinical and speech features
df_full = pd.concat([df_demographic_clinical_feats, df_speech], axis=1)
df_full = df_full.drop(['dep_scale'], axis = 1)

#drop features that are highly correlated
df_full = df_full.drop(columns = to_drop)
# df_speech = df_speech.drop(columns = to_drop)
# df_demographic_clinical = df_demographic_clinical_feats.drop(columns = to_drop)

df_full = df_full
df_speech = df_speech
df_demographic_clinical = df_demographic_clinical_feats
```

Splitting the dataset

```
# create target label
df_target = df.depressed

# split our data
X_train, X_test, y_train, y_test = train_test_split(df_full, df_target, test_size=0.2, random_state=10)
X_train_speech, X_test_speech, y_train_speech, y_test_speech = train_test_split(df_speech, df_target, te

# summarize the shape of the training datasets
print(X_train.shape, y_train.shape)
print(X_train_speech.shape, y_train_speech.shape)

(95, 206) (95,)
(95, 192) (95,)
```

Splitting the dataset

# Fitting the Models: Outlier Detection

- Next, we checked data distribution and outliers in the speech features.
- As we have many features, we needed to have an automated way to determine whether or not the features contain skewed distributions and if they contain any outliers. We used **Isolation Forests** for detecting outliers.

```
#  identify outliers in the training dataset
iso = IsolationForest(contamination=0.05)
yhat = iso.fit_predict(X_train)

# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train.values[mask, :], y_train.values[mask]
X_train_speech, y_train_speech = X_train_speech.values[mask, :], y_train_speech.values[mask]

# summarize the shape of the updated training dataset
print(X_train.shape, y_train.shape)
print(X_train_speech.shape, y_train_speech.shape)
```

```
(90, 206) (90,)
(90, 210) (90,)
```

Outlier detection with Isolation Forests

# Fitting Models: Baseline Model

- Our baseline model is the **Support Vector Machine** (SVM) classifier. We first fit the full dataset (both speech and non-speech features).

```
#Create a svm Classifier
clf = SVC(kernel='linear') # Linear Kernel

# start time
start = time.time()

#Train the model using the training sets
clf.fit(X_train, y_train)

# check training time
print("Train time: {0}".format(time.time() - start))

_model_SVM_dur = time.time() - start

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
Train time: 0.0029196739196777344
```

```
# Model Accuracy: how often is the classifier correct?
print("Accuracy:", accuracy_score(y_test, y_pred))

# get the accuracy score for the plot
_model_SVM_acc = accuracy_score(y_test, y_pred)
```

```
Accuracy: 0.6666666666666666
```

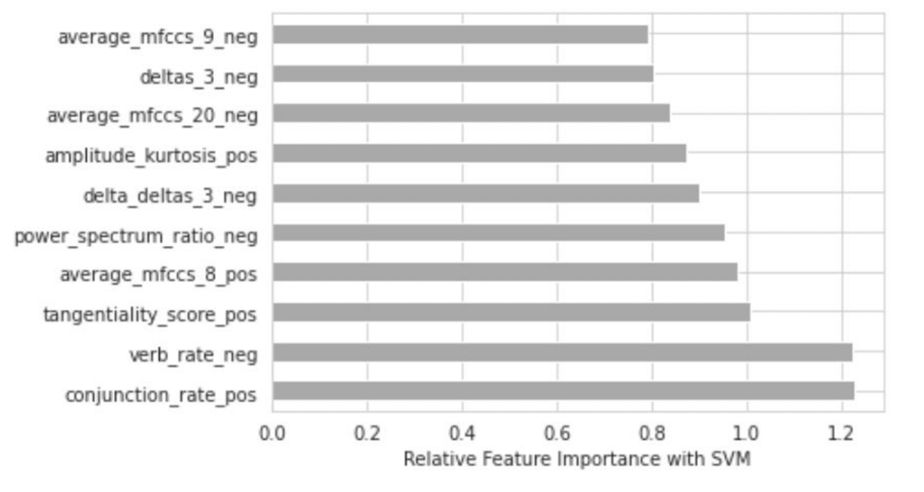
```
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:", precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:", recall_score(y_test, y_pred))
```

```
Precision: 0.7
Recall: 0.5833333333333334
```

# Fitting Models: Baseline Model

- As we can see **conjuction_rate_pos**, **verb_rate_neg** are the most contributing features. These variables deal with the frequency of the verbs in the speech and the conjunction rate in speech.




Feature importance for the SVM

# Fitting Models: Optimizing Hyper-Parameters

- In order to optimize the results of the model, we used a technique called ‘cross-validation’.
- Cross-validation splits up the training and test sets into k splits in order to find the arrangement that yields the best results. We used a 5-fold cross validation split in our model.


Cross-validation

# Fitting Models: Optimizing Hyper-Parameters

- Another technique we used to optimize the results is **hyperparameter tuning.** We specified a testing range for all of the parameters that the SVM model takes and try out all the different permutations in order to find the best combination.
- Cross-fold validation as well as hyperparameter-tuning can be implemented in a single line using the ‘GridSearchCV’ function from the sklearn library. We also specified that we want to optimize the precision and recall of the model.

```
# Set the parameters by cross-validation
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

scores = ['precision', 'recall']
```

Hyperparameters

```
clf = GridSearchCV(
    SVC(), tuned_parameters, scoring='%s_macro' % score
)
clf.fit(X_train, y_train)
```

Fitting grid search

# Fitting Models: Results

- Training the model with the full dataset (both speech and non-speech features) gives us these results.
- We can clearly see that there is some room for improvement.

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

         0.0       0.62      0.67      0.64        12
         1.0       0.64      0.58      0.61        12

    accuracy                           0.62        24
   macro avg       0.63      0.62      0.62        24
weighted avg       0.63      0.62      0.62        24
```

Classification report

# Fitting Models: Results

Training the SVM model with the speech variables only gives us these results. The accuracy dropped slightly comparing to the model trained with full dataset.

```
#Create a svm Classifier
clf_speech = SVC(C= 1, kernel='linear') # Linear Kernel

# start time
start = time.time()

#Train the model using the training sets
clf_speech.fit(X_train_speech, y_train_speech)

# check training time
print("Train time: {0}".format(time.time() - start))

#Predict the response for test dataset
y_pred_speech = clf_speech.predict(X_test_speech)
```

```
Train time: 0.0020954608917236333
```

```
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",accuracy_score(y_test_speech, y_pred_speech))
```

```
Accuracy: 0.5833333333333334
```

```
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:", precision_score(y_test_speech, y_pred_speech))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:", recall_score(y_test_speech, y_pred_speech))
```

```
Precision: 0.5714285714285714
Recall: 0.6666666666666666
```

# Fitting Models: Other Models

- We also experimented with other models: XGBoost Classifier, K-NN classifier, Deep-learning with Keras (The Sequential Model), Bayesian Network, Naive Bayes, Decision Tree Classifier
- We trained on the **full dataset** to compare the training time and accuracy.

| | Accuracy | Training Time |
|---|---|---|
| SVM model | 0.6666666666666666 | 0.0030972957611083984 |
| K nearest-neighbour | 0.5833333333333334 | 0.0007801055908203125 |
| KNN hyper_tuned | 0.5888888888888889 | 1.205317497253418 |
| KNN fold cross validation | 0.5555555555555556 | 0.08338260650634766 |
| Keras NN | 0.5833333134651184 | 8.289164781570435 |
| Naive Bayesian | 0.625 | 0.0024428367614746094 |
| Decision Tree | 0.3333333333333333 | 0.006578683853149414 |
| XGboost | 0.5 | 0.18633222579956055 |

Comparing models

# Fitting Models: Results

- Training the model with our newly created dataset using all the optimisation techniques gives us the following results.
- Comparing them to the results of the other models we tried we notice that these are the best by far, and thus we can confidently claim that the results are maximized and that the SVM model is indeed the best for our task of depression classification, which is nothing else than simple binary classification on a speech-features dataset.

# Fitting Models: Gender Differences

Next, we checked how our baseline SVM model predicts the depression between male and female based on speech features only. We split the dataset on training and testing sets. The first dataset contains only **female observations.** The second dataset contains only **male observations**.

```
[66]
# split our data in female and male
dfs = [rows for _, rows in df.groupby('gender')]

# create dataframe for female
df_speech_female = dfs[0]
# create dataframe for male
df_speech_male = dfs[1]

df_speech_male = df_speech_male.drop(columns = demographic_clinical_feats)
df_speech_female = df_speech_female.drop(columns = demographic_clinical_feats)

df_speech_male_target = df_speech_male.depressed
df_speech_female_target = df_speech_female.depressed

df_speech_male = df_speech_male.drop(['depressed'], axis = 1)
df_speech_female = df_speech_female.drop(['depressed'], axis = 1)

X_train_speech_male, X_test_speech_male, y_train_speech_male, y_test_speech_male = train_test_split(df_speech_male, df_speech_mal
X_train_speech_female, X_test_speech_female, y_train_speech_female, y_test_speech_female = train_test_split(df_speech_female, df_

# summarize the shape of the training datasets
print(X_train_speech_male.shape, y_train_speech_male.shape)
print(X_train_speech_female.shape, y_train_speech_female.shape)
```

```
(21, 210) (21,)
(73, 210) (73,)
```

# Fitting Models: Gender Differences

- After splitting the dataset, we only have **21 male** participants and **73 female** participants.
- For predicting the depression for each of the genders, we will not detect outliers and remove highly correlated features, as we don't have that many observations for each of the participants.

# Fitting Models: Gender Differences

The accuracy for the male participants is very low. This is most likely due to the fact that we have very few observations of the male subjects.

```
# Model Accuracy: how often is the classifier correct?
print("Accuracy:", accuracy_score(y_test_speech_male, y_pred_speech_male))
```

```
Accuracy: 0.3333333333333333
```

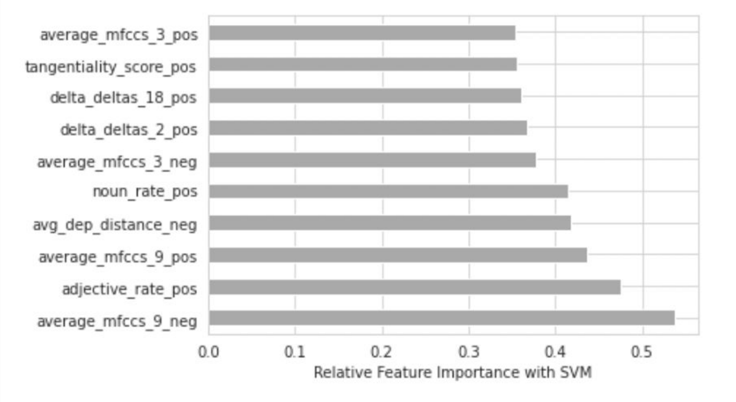
```
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:", precision_score(y_test_speech_male, y_pred_speech_male))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:", recall_score(y_test_speech_male, y_pred_speech_male))
```

```
Precision: 0.5
Recall: 0.5
```

# Fitting Models: Gender Differences

The **average_mfccs_9_neg**, and **adjective_rate_pos** are the most contributing features for the male participants.




Feature importance for males

# Fitting Models: Gender Differences

Training on the female dataset gives us better results than male.

```
# Model Accuracy: how often is the classifier correct?
print("Accuracy:", accuracy_score(y_test_speech_female, y_pred_speech_female))
```

```
Accuracy: 0.5789473684210527
```

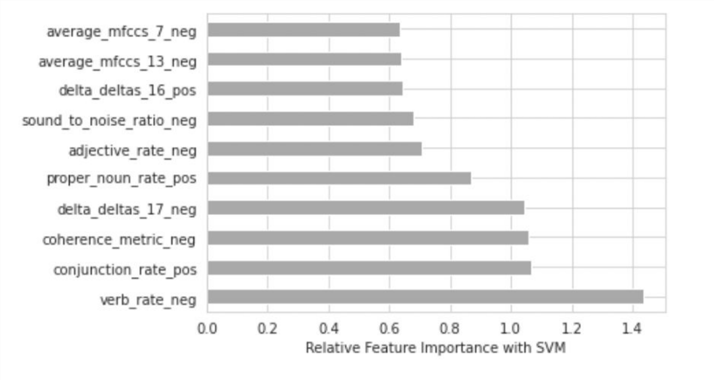
```
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:", precision_score(y_test_speech_female, y_pred_speech_female))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:", recall_score(y_test_speech_female, y_pred_speech_female))
```

```
Precision: 0.6666666666666666
Recall: 0.5454545454545454
```

# Fitting Models: Gender Differences

The **verb_rate_neg,** and **conjuction_rate_pos** are the most contributing features for the female participants.




Feature importance for females

# Fitting Models: Gender Differences

Next, we try to take the 10 most important features for both genders and see if the accuracy would improve.

```
df_female_coef = pd.Series(abs(clf_speech_female.coef_[0]), index=df_speech_female.columns).nlargest(10)

df_female_features = pd.DataFrame(df_speech_female, columns=df_female_coef.index)
df_female_coef
```

```
verb_rate_neg             1.434914
conjunction_rate_pos      1.068183
coherence_metric_neg      1.056574
delta_deltas_17_neg       1.046028
proper_noun_rate_pos      0.866702
adjective_rate_neg        0.705109
sound_to_noise_ratio_neg  0.680327
delta_deltas_16_pos       0.642284
average_mfccs_13_neg      0.639014
average_mfccs_7_neg       0.636509
```

Most important features for females

```
df_male_coef = pd.Series(abs(clf_speech_male.coef_[0]), index=df_speech_male.columns).nlargest(10)

df_male_features = pd.DataFrame(df_speech_male, columns=df_male_coef.index)
df_male_features
```

| | average_mfccs_9_n… float… 0.0766496730054923 – 1.0 | adjective_rate_pos float64 0.0 – 1.0 | average_mfccs_9_p… float… 0.2513971097667385 – 1.0 | avg_dep_distance_… float… 0.15802247922167578 – 0.84… |
|---|---|---|---|---|
| 3 | 0.5916265382099758 | 0.589285714622449 | 0.46232776676127907 | 0.6871049540064338 |
| 4 | 0.7550010175022297 | 0.4489795893994169 | 0.6916732028457038 | 0.22562933094214654 |
| 5 | 0.9260310989245442 | 0.6132404209218516 | 0.8218791679590939 | 0.7775416191042144 |
| 6 | 0.8161387311538182 | 0.333754740190717 | 0.7270792235841967 | 0.7191242446252832 |
| 8 | 0.8640729095121004 | 1 | 0.8076591976064239 | 0.6423024744418413 |

Most important features for males

# Fitting Models: Gender Differences

As we can see, the accuracy for the male participants is the same when taking only the ten most contributing features. However, the accuracy for the female participants has improved.

Train time: 0.0021309852600097656
Accuracy: 0.3333333333333333

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.33 | 1.00 | 0.50 | 2 |
| 1.0 | 0.00 | 0.00 | 0.00 | 4 |
| accuracy | | | 0.33 | 6 |
| macro avg | 0.17 | 0.50 | 0.25 | 6 |
| weighted avg | 0.11 | 0.33 | 0.17 | 6 |

Classification report for males

Train time: 0.35413455963134766
Accuracy: 0.6842105263157895

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.62 | 0.62 | 0.62 | 8 |
| 1.0 | 0.73 | 0.73 | 0.73 | 11 |
| accuracy | | | 0.68 | 19 |
| macro avg | 0.68 | 0.68 | 0.68 | 19 |
| weighted avg | 0.68 | 0.68 | 0.68 | 19 |

Classification report for females

# Outlook

- Experiment with different combinations of clinical, demographic and speech features.
- Improving performance by using different **preprocessing** and **feature extraction** methods (e.g., finding the most contributing speech features, removing collinear speech features from a specific gender)
- Collecting a **larger dataset** to improve the accuracy and generalizability of the model.
- Consider positive and negative events separately for both genders.

# Any Questions?