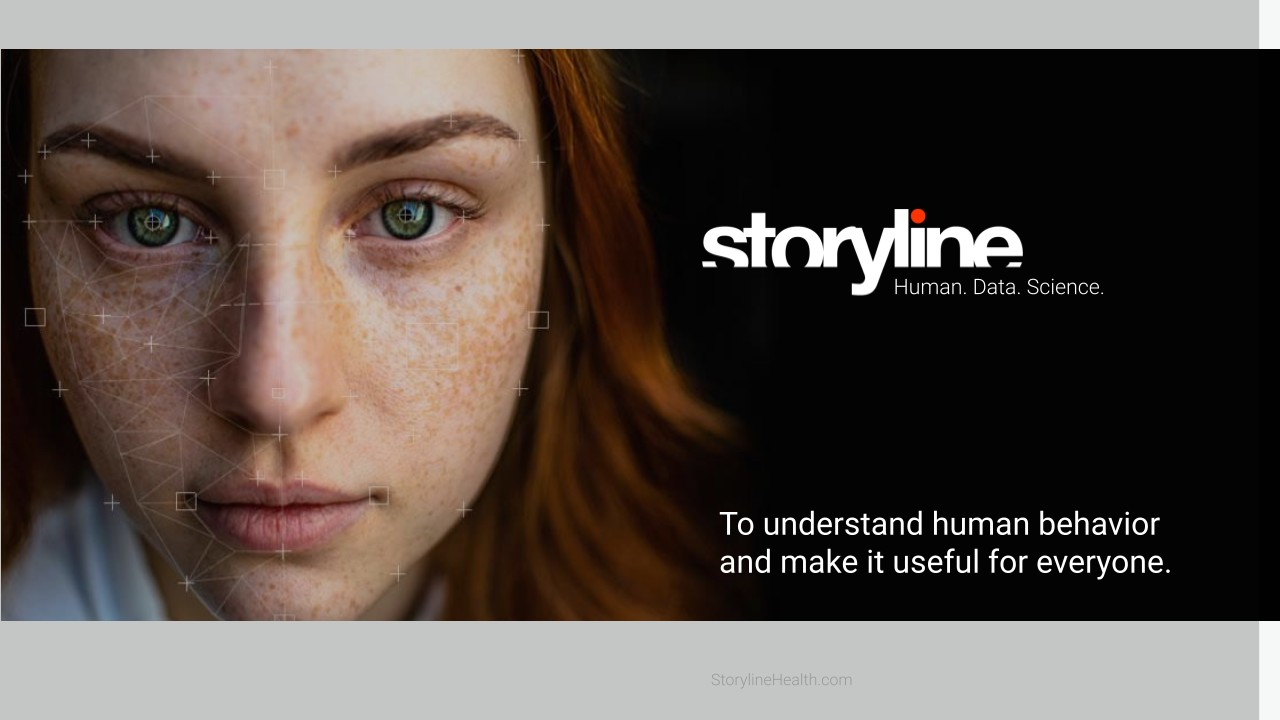
|  |
| --- |
| Anxiety Disorder Detection |
|  |
| October 5  COMPANY NAME  Storyline  Authored by: Muzammil Ahmed Shaik |



# OVERVIEW

|  |
| --- |
| Objective Prediction of individuals with symptoms of a clinical anxiety disorder from Storyline facial, speech, and, vocal data collected from 100 people throughout the United States through a Storyline smartphone interview.  The data contains the patient ID numbers of everyone in the study in the first column. The other 79404 columns are populated with measures of facial, speech, and vocal patterns expressed by the study participants during their Storyline interview. |
|  |
| Fig 1a. First four rows and 6 columns of the data |
| **Target Variable**  Anxiety  This file contains the clinical diagnoses of the study participants. The first column indicates the patient ID numbers for everyone in the study. The other columns indicate the diagnosis for everyone for anxiety (column 2) and other mental health concerns as TRUE or FALSE. This project only focuses on the anxiety diagnosis in column 2. |

Fig 2a. first 5 rows and a column of the response variable

There are 3 values in this column:

True: The patient suffers from an anxiety disorder.

False: The patient doesn’t suffer from an anxiety disorder.

NaN: Missing value.

The most alarming value here is NaN and our first step in this project is to handle these missing values.

Feature Engineering

Handling Missing values

a) Dropping features

b) Fill in the column with Mean, Median and Mode

c) The other methods which aren’t used in this project

Features with missing values:

Text

Description automatically generated

Fig 3a. Number of missing values in the data

As there are only 3 values in each feature its not good to drop those rows or features and hence, we will be using mean, median and mode to fill in the missing value.

a) Mean, median and mode

This method sounds quiet confusing as these three statistical methods are of different methodologies and how they can be implemented together? Yes, it is difficult to implement all three methods together, but we will be using one of the methods to fill in the missing value depending on the skewness of the feature.

Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.

Skewness determines which method to be implemented to fill in the missing values and for finding this skewness we go through next step of the project and that is Data Visualization.

If the data is skewed then we can not use mean as the method to fill in the missing values, instead we can use median or mode to fill in the missing values.

Data Visualization

One of the features(agreeableness\_percentile\_q1) has been plotted to find skewness.

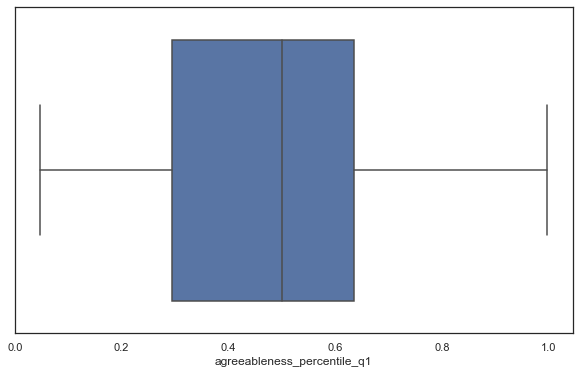


Fig 4a. Boxplot for agreeableness\_percentile\_q1 feature

Chart, histogram

Description automatically generated

Fig 4b. Distribution plot for agreeableness\_percentile\_q1 feature

We can clearly see that feature is positively skewed and we will not be using mean to fill in the values for this feature, instead median or mode of the feature can be filled in place of missing values.

To confirm the method to be used to fill in the missing values of the data, we plot more features to find their skewness and if the data is skewed then we will not be using mean method to fill the missing data.

With missing values in the features there were some infinite values which were concerning for data analysis and building model. To handle that we fill the infinite values with zero’s so that they should be normalized and doesn’t affect the model.

Second feature is sympathy\_percentile\_q1 to find skewness.

Chart, box and whisker chart

Description automatically generated

Fig 4c. Boxplot for sympathy\_percentile\_q1 feature

Chart, histogram

Description automatically generated

Fig 4d. Distribution plot for sympathy\_percentile\_q1

Even this plot shows that data is positively skewed and mean can not be used to fill in the missing value. Median was used to fill in the missing values for the whole dataset, other features were also plotted and checked for skewness to know which method to use. Finally, median was used to fill in the input data and output data was filled with mode.

Text

Description automatically generated

Fig 4e. Data after filling missing values with median

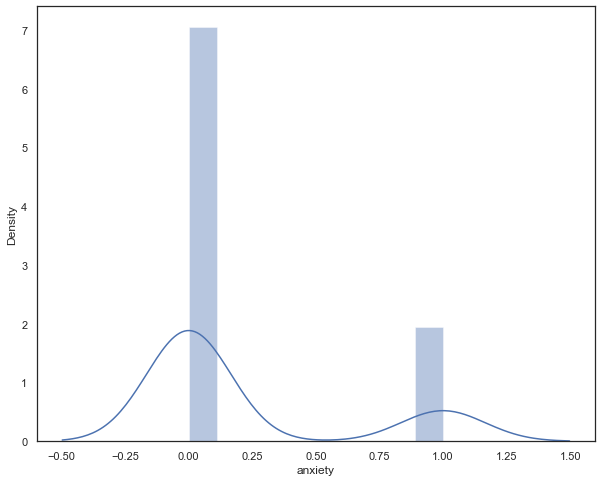


Fig 4f. Distribution plot for anxiety feature

Anxiety is also skewed so we used mode method to fill in the missing value and infinite values were turned to zero.

Graphical user interface, application, Teams

Description automatically generated

Fig 4g. Anxiety feature after filling the missing values

Number of true and false were calculated to get overview of the anxiety variable.

Chart, bar chart

Description automatically generated

Fig 4h. Number of True and False in Anxiety feature

Feature Transformation

The process of transforming raw data into reduced format so that it represents the underlying problem to the predictive models.

array([[-1.49578908, -1.94159541, -3.61832956, ..., -0.88680261,

-0.94868742, -0.46015012],

[-0.61717489, 0.06570655, -0.34034036, ..., 0.69397855,

-1.22215842, -0.66315559],

[-1.96121226, -0.42171963, 0.59784743, ..., 1.03162918,

-1.68239442, 0.38158823],

...,

[ 0.08528132, 0.80059778, 0.72112467, ..., 0.01324099,

-0.61540254, 0.45267657],

[ 0.35459004, -0.12520158, -0.96043773, ..., 0.94962726,

-0.0552211 , 1.30860864],

[-1.01051004, -0.52295314, -0.48467735, ..., 0.02931911,

1.0362754 , -0.11783856]])

Chart, histogram

Description automatically generated

Fig 5a. Standardization of the data

Feature Engineering

Violin Plot

This plot is used to know how features are related with anxiety with respect to median. Plotting features with respect to anxiety.

A picture containing arrow

Description automatically generated

Fig 6a. Violin plot for knowing median differences for first 10 features

A picture containing arrow

Description automatically generated

Fig 6b. Violin plot for next 10 features

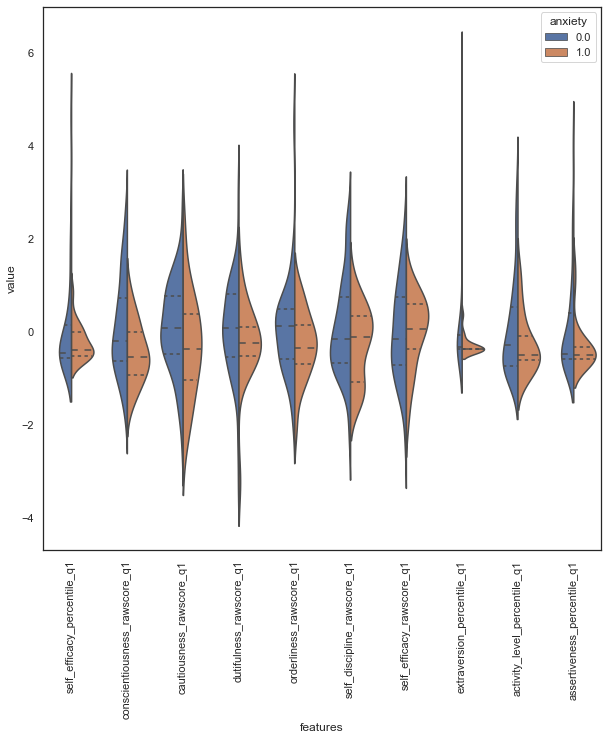


Fig 6c. Violin plot for next 10 features

From above plots which contains information about features from 1-30 none of them shows any difference in the median and it is difficult to plot all(~80k) the features and observe the median differences with respect to anxiety.

This median difference can also be observed using boxplot.

Boxplot

This median difference can also be observed using boxplot.

Chart, box and whisker chart

Description automatically generated

Fig 7a. Boxplot for knowing median differences for first 10 features

Even this plot shows that there are no median differences in the features to be considered for classification.

Joint Plot

With boxplot we can see that activity\_level\_percentile\_q1 and assertiveness\_percentile\_q1 are related with each other. To know correlation between this features we can clearly see-through joint plot.

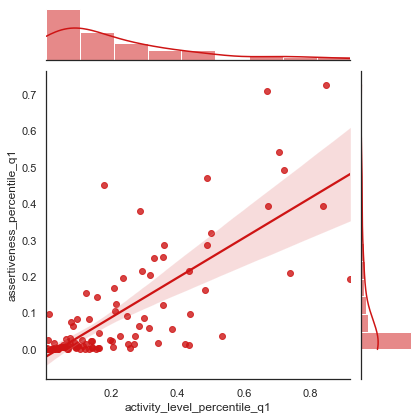


Fig 8a. Joint plot for activity\_level\_percentile\_q1 and assertiveness\_percentile\_q1

From this plot we can clearly how these two features are related with each other

Now let us compare three features together and try to know correlation between them using pair-grid plot.

Pair-Grid Plot

This plot compares three features together to know correlation between them.

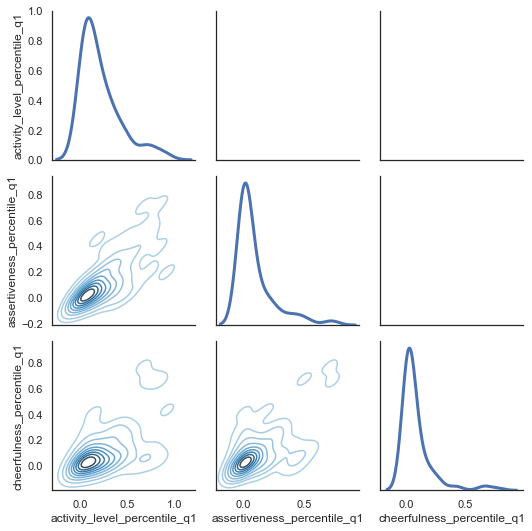


Fig 9a. Pair-grid plot for activity\_level\_percentile\_q1, assertiveness\_percentile\_q1 and cheerfullness\_percentile\_q1

Using pair grid plot we can clearly see how these three features are related with each other.

The main problem with this project is number of features which are approximately 80k and plotting this feature and analyzing them is not possible.

Highly prioritized task is to handle this feature and reduce them, this task is known as feature selection, one of the major challenges faced in this project was how to handle these features and how to drop them. Normally a PC or laptop contains 4GB of ram and when doing feature selection, it was asking for 46 GB which is 10 times greater than what an average pc has.

Before jumping into feature selection, it was extremely critical job to handle memory efficiently.

Downcast

Downcast is a method to change datatypes of the features to a reduced format to efficiently use memory and it also help us to deal with all the features.

Main idea behind this technique is to format the data types of the features, for example if a column is in float but it has number which can also be represented in integer then it converts that column to integer which internally means a 16 bit has been changed to 8 bit or even 4 bits. This drastic change in bits will help us to reduce memory usage for feature selection and running all other functions like building models inexpensively.

Text, letter

Description automatically generated

Fig 10a. Function for Down casting datatypes

The above function helps us to downcast the features containing data frame.

As the memory is now ready for usage, we were ready with feature selection techniques to reduce the features. Handling ~80k features and reducing them to 10 was noteworthy achievement to build model for predicting anxiety.

Feature Selection

Random forest feature importance method was used to find the score of each feature, this score is based on how feature is related with anxiety and how much is it important for the model.

Graphical user interface, text, application, email

Description automatically generated

Fig 11a. Random Forest feature importance score calculation model

Average score was calculated for the dataset, and it turns out to be very low that is 0.00001. Keeping the features which has this much score would overwork the model and decrease our accuracy of the model and would also increases number of features to be considered which is not an ideal condition for a model.

We decided to drop the features which has score less than the average score, to implement this we have taken 2 steps.

First, we created empty list to fill out the features which meets the condition.

Secondly, we created the new data frame to store those features which have met the criteria.

These two steps are demonstrated below with the code and its output. This method helped us to reduce features from ~80k to 634. Wow! this method was super-efficient to get rid of waste columns from the data.

Text

Description automatically generated

Fig 11b. Creating the list for features that met the average condition

Text

Description automatically generated with low confidence

Fig 11c. Created new data frame with features meeting the average score condition.

Correlation Matrix

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

Finding correlation between features is not a difficult task but dropping one of the features and making sure only one features stays in the model is a tricky task. For that we have created a function which finds only one pair and return us that feature to drop it.

Text

Description automatically generated

Fig 12a. Confusion matrix for all the features remained after meeting average score criteria

Using the heat map from the above figure we can clearly see that there are some features which are correlated with each other and needs to be dropped.

Table

Description automatically generated

Fig 12b. Correlated features dropping code

This method helped us to drop the features from 634 to 517, still we need to do more feature selection to get best model for anxiety prediction.

Select-K-Features

This method helps us to select k best features from the model provide mutual information of the data.

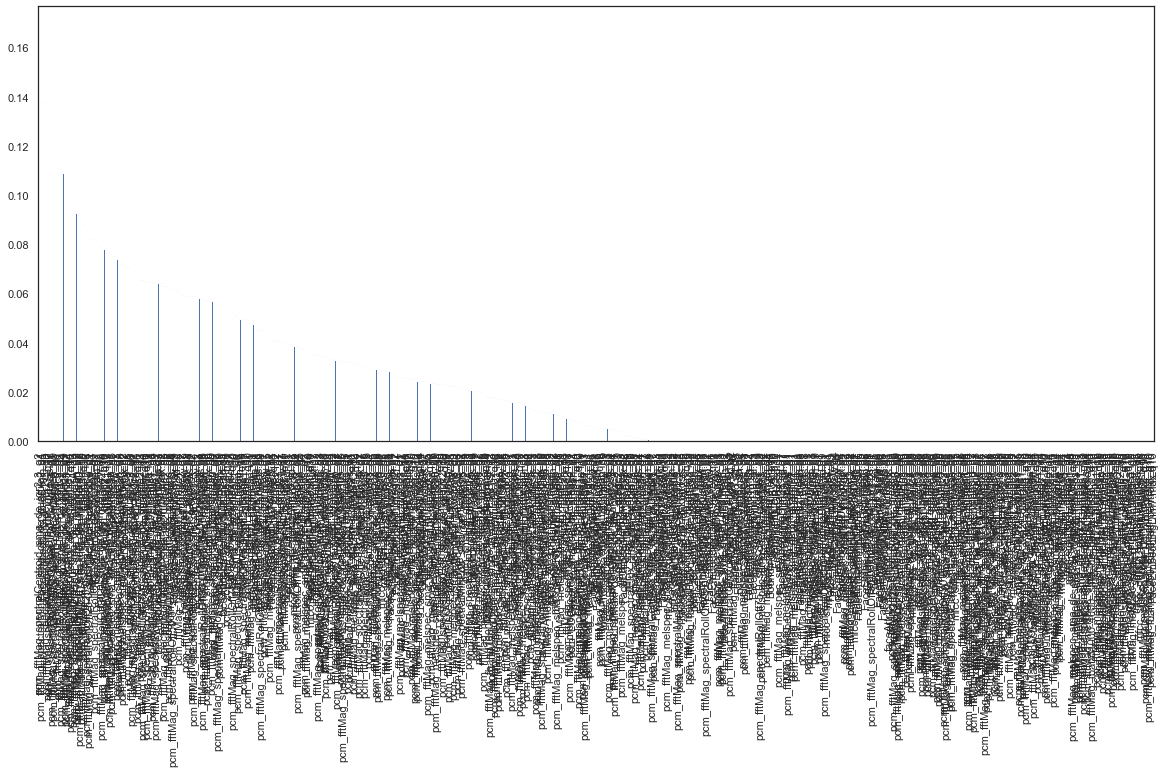


Fig 13a. Bar graph all features providing information related to anxiety

This figure clearly shows that there are only 21 features which are providing us the information and even in that there are only 10 features which are providing information greater than 0.05 so we select those 10 features for the anxiety prediction.

Fisher’s Exact test p-value

Top 10 features selected from ~80k features using select-k-best method are:

a) pcm\_fftMag\_spectralRollOff90.0\_sma\_linregc2\_q1

b) pcm\_fftMag\_spectralRollOff50.0\_sma\_de\_de\_iqr1-3\_q1

c) pcm\_fftMag\_spectralMinPos\_sma\_de\_de\_quartile2\_q1

d) pcm\_fftMag\_spectralCentroid\_sma\_de\_iqr2-3\_q2

e) mfcc\_sma\_de[6]\_percentile95.0\_q6

f) pcm\_fftMag\_spectralRollOff50.0\_sma\_de\_quartile3\_q9

g) pcm\_fftMag\_spectralCentroid\_sma\_de\_iqr2-3\_q9

h) pcm\_fftMag\_spectralCentroid\_sma\_de\_iqr1-3\_q10

i) FaceEmbeddingDim66.min.q1

k) FaceEmbeddingDim101.kurtosis.q7

After selection of this features, we might think of dropping more features or are this feature significant for the model? This all questions can be answered using Fisher’s exact test p-value. If the p-value is less than 0.05, then that feature is significant.

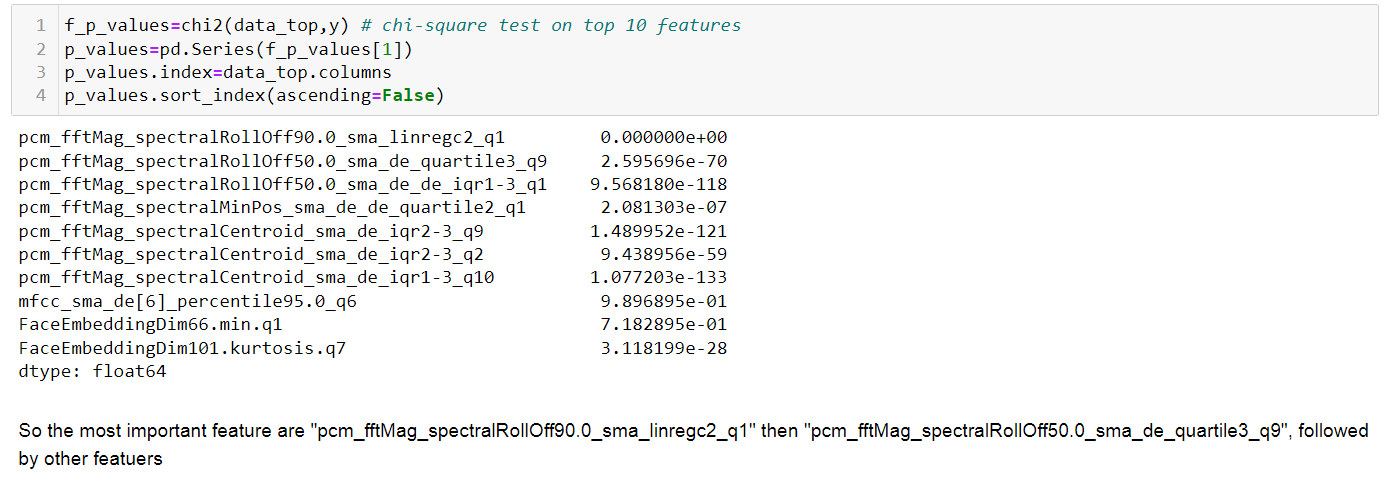


Fig 14a. Chi-square test for top 10 features

Seeing the P-values of the top 10 features, we can say that all the features listed below are significant and none of them can be dropped from the dataset.

Finally, we were able to achieve 10 features from ~80k features which is excessive triumph for building a model. Ideally a good model has 6-8 features as it is difficult to monitor features more than 10 but 10 is however adequate.

Machine Learning Models

Ranking of models based on the accuracy:

1) Bagging Decision Tree: Accuracy = 92%

score on test: 0.92

score on train: 0.9466666666666667

Bagging Decision Tree is an optimal model.

2.a) K-Nearest-Neighbors: Accuracy = 84%

score on test: 0.84

score on train: 0.8266666666666667

KNN is an optimal model.

2.b) Random Forest: Accuracy = 84%

score on test: 0.84

score on train: 1.0

Random Forest is an overfitting model.

2.c) Voting Classifier: Accuracy = 84%

score on test: 0.84

score on train: 0.9333333333333333

Voting Classifier is an overfitting model.

3.a) Boosting Decision Tree: Accuracy = 76%

score on test: 0.76

score on train: 1.0

Boosting Decision Tree is an overfitting model.

3.b) Neural Network (Tuning): Accuracy = 76%

Testing accuracy: 0.76

Training accuracy: 0.8133333333333334

Neural network is an optimal model

3.c) XG Boost Tree: Accuracy = 76%

score on test: 0.76

score on train: 1.0

XG boost Tree is an overfitting model

4.a) Logistic Regression: Accuracy = 72%

score on test: 0.72

score on train: 0.8666666666666667

Logistic Regression is an overfitting model.

4.b) Support Vector Machine: Accuracy = 72%

score on test: 0.72

score on train: 0.8

Support Vector Machine model is an optimal model

4.c) Decision Tree: Accuracy = 72%

score on test: 0.72

score on train: 1.0

Decision Tree is an overfitting model

5.a) Multinomial Naive Bayes algorithm: Accuracy = 44%

score on test: 0.44

score on train: 0.68

Multinomial Naive Bayes model is an underfitting model

Out of these models, the best, optimal , and most accurate model is bagging decision tree which gave 92% accuracy to predict symptoms of anxiety of the patient.

Confusion Matrix

a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.

Chart

Description automatically generated

Fig 16a. Confusion matrix with necessary data

This confusion matrix is plotted for Bagging decision tree model which has achieved 92% accuracy in predicting anxiety symptoms when facial and vocal data is given.

The features that have been selected in the model are related to vocal data converted to digits, voice data converted to digits and facial expression embedded to numbers.

Given these features, using bagging decision tree as a model we can predict whether a patient suffers from anxiety disorder. In real world, a doctor or more precisely a psychiatrist will be able to diagnose anxiety disorder using these 10 features that we have selected for the model. The other features which have not included in this model and decided to drop them for the prediction were sensible and reasonable.

Future Work

“Everything in this world is replaceable, you just need to find the perfect fit”

* Data analysis can be further extended using high end pc, memory usage on this kind of pc’s would be enough for running such kind of huge data.
* Visualization of the data can be done taking many features at a time rather than 10 or 20 at once to find relation between them and the target variable.
* Feature selection can be done using forward and backward selection making R^2 or AIC or BIC as the criteria for selection.
* Standardization and normalization can be implemented in the model building to get more efficient accuracy.
* Hyper parameter tunning of Bagging Decision Tree can be done to increase the accuracy
* Pipeline methods can be implemented with deep learning network and other best models which gave excellent accuracy results.