Machine Learning Models for Speaker Personality Prediction from Speech Signals

Toni Ivanov

Bachelor's Thesis
Department of Computer Science
University of Crete

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Advisor: Prof. Yannis Stylianou
Supervisor: Dr George P. Kafentzis

December 5, 2024

Contents

List of Figures

List of Tables

Abstract

In Psychology, Personality Assessment involves the systematic measurement and evaluation of enduring and distinctive patterns of cognition, emotion, behavior, and interpersonal functioning that characterize individuals. It is conducted through various validated methods and instruments designed to provide a quantifiable and reliable understanding of an individual's personality traits and attributes.

However, in Human-Computer Interaction assessing someone' s personality is still a challenging task. While there are text, image, and speech based methods for this task, this thesis provides a framework for utilizing machine learning (ML) methods to study the connection between personality traits and speech features.

The ML pipeline includes feature extraction, feature selection, classification, and classification report. Features include spectral and temporal characteristics such as MFCCs, pitch, zero-crossing rate, formants, and common spectral measures, among others. Feature selection involves Recursive Feature Selection, K-Best features, and PCA. Selected machine learning algorithms for classification include weak learners, such as Logistic Regression, Support Vector Machines, Naive Bayes, Decision Trees, and kNNs, and strong learners like Random Forests and boosting models (AdaBoost, XGBoost).

The SSPNet corpus is used to extract speech features and to map them to personality traits according to the Big-5 categorization (Openness, Conscientiousness, Extroversion, Agreeableness, Neuroticism). Each trait is split in two classes, a "high" and a "low", making the task a binary classification problem, significantly simplifying the process. A classifier is trained for each trait separately. Training includes hyperparameter tuning using a K-fold Cross-Validation strategy.

The system we provide offers a User Interface (UI) that lets a user specify a classification algorithm, a feature selection method, and a scoring function for the hyperparameter optimization. Once the model is trained and tested on the dataset, information regarding classification performance is returned to the user in order to assess the predicting capabilities of the modeling pipeline. Classification report includes a Confusion Matrix, a Receiver Operating Curve (ROC) with its Area Under Curve (AUC), and a Precision-Recall-F1score report.

Chapter 1

Introduction

Speech is often seen as a reflection of personal expression. Through speech, we convey emotions. It may be regarded as a form of gesture. When we are expressing a thought, a feeling, we may use our hands or some other type of gesture, and our speech takes part in the complete arrangement of the expression. According to the *Brunswik Lens* cognitive model, we externalize our personality through distal cues, which encompass speech among other factors. On the other hand, when an observer receives these cues, the cues initiate a process of perception that leads in the creation of a percept, i.e. the mental representation of something that is perceived. The cues that the observer perceives are called proximal cues and are distinguished from the distal cues. For example, the listener does not perceive the vibrations in the speaker's vocal chords, but rather interprets the tone and emotion of the speaker's voice. Ultimately, the observer attributes a specific personality based on this percept. The assessment of perceivable manifestations of personality from speech is the focus of this work. This is a prosody approach to *Automatic Personality Perception* (APP). The goal of APP is to study the personality perceived by observers. When studying personality in terms of speech, the focus shifts from the overall personality-such as forming judgments from past events-to concentrating on observable traits. This results to traits that can be noticed right when people talk during an experiment. The traits should be easy to understand by anyone, even if they don't know the speaker.

There are several reasons to recognize the importance of the problem. Computer agents are now an integral part of our daily lives. In Nass et al.[1], the authors report that people will respond to the personalities computer agents emit, in the same way they would respond to similar human personalities. In a study involving 48 participants, dominant and submissive individuals were matched with dominant or submissive computers.The computers exhibited traits associated with dominance or submissiveness. Participants recognized the computer's personality type and preferred interacting with computers that shared their personality trait, showing that even minimal cues can influence computer personalities and human responses.

Furthermore, studying the results of Automatic Personality Perception helps us understand how people are influenced by our behavior. *The meaning of many social events is constructed routinely, habitually, and unintentionally i.e., spontaneously* [2]. "Spontaneous trait inferences" (STIs) occur when we interpret someone's behavior to infer their traits without consciously intending to do so. In our case the behavior is the prosodic features of speech that we display. For example, during a job interview, if a candidate speaks confidently and articulately, the interviewer might infer traits like competence and strong communication skills. Moreover, call center operators rely heavily on speech cues to infer customer emotions and needs. If a customer talks calmly and politely, the operator might think the customer is patient and easy to work with. Therefore, it is important to study the traits that are reflected to an observer through non verbal cues.

1.1 Review of Related Studies

One early study was conducted by Allport and Cantril [3]. They conducted ten experiments at the Harvard Psychological Laboratory and the broadcasting studio of Station WEEI in Boston where eighteen male speakers and over six hundred judges took part. The primary methodology involved correlating objective personality information with the corresponding vocal characteristics of the speakers. One primary conclusion is that there is a noticeable level of agreement among listeners when it comes to judging personalities from voices, but these judgments might not always align perfectly with the true personalities of the speakers.

Addington [4] conducted a research to understand the impact of voices in eliciting stereotyped personality judgments. Two hundred and fifty-two tape-recorded samples of the "Rainbow Passage" were obtained from two male and two female trained speakers instructed to simulate seven voice qualities and three variations of speaking rate. Two sets of data was collected. Vocal data were collected from three groups of trained judges asked to describe the vocal characteristics of the samples. The second set of data, perceived personality data, was

obtained from naive subjects who described the personality of the speakers. He concluded that that judgments of listeners ascribing personality from samples of speakers' voices tend to be uniform. The results also showed that although he perceptions of males and females do not differ appreciably, the perceptions engendered by male and female speakers do differ according to the vocal characteristic simulated.

Zuckerman and Driver [5] focused on the premise that individual differences in vocal attractiveness may elicit different impressions of personality. The effects were obtained in two studies using somewhat different methods. In one study, judges rated senders reading a standard-content speech; in the other, judges rated senders reading randomly selected paragraphs. First, observers were able to agree on what is vocally attractive. Second, senders with more attractive voices were rated more favorably than senders with less attractive voices.

Attempts have been made to study the problem of perception of persons from the nonverbal properties of their speech experimentally by utilizing techniques of speech synthesis by computer. Brown et al. [6] analyzed and synthesized by an automatic speech analysis-synthesis scheme the voices of two adult male college teachers speaking the sentence "We were away a year ago". Each voice was synthesized in 27 forms (all possible combinations of three values of each of mean $F0$ level, rate, and variance of $F0$). The tape was played to a group of 37 male and female judges who rated the voices on 15 adjectives. They found that lowering the speech rate resulted in a drastic loss of perceptual competence and a moderate loss of benevolence. Increasing the speech rate acted vice versa. As a less strong effect, increasing pitch variability raised the impression of increased benevolence while decreasing it also caused perceived competence to decrease. Finally, an increased pitch mean caused both low competence and low benevolence.

Mohammadi and Vinciarelli [7] proposed an approach for the automatic prediction of the traits the listeners attribute to a speaker they never heard before. They used a corpus (SSPNet) of 640 speech clips annotated in terms of personality traits by 11 assessors. The results show that it is possible to predict with high accuracy (more than 70% depending on the particular trait) whether a person is perceived to be in the upper or lower part of the scales corresponding to each of the Big-Five, the personality dimensions known to capture most of the individual differences.

1.1.1 Our work

In this work we provide a Python-based platform for predicting personality traits from speech signals. We use the SSPNet corpus consisting of 640 speech clips. Then each speech clip is categorized as either HIGH or LOW based on the average assessments of personality traits by the judges. Features and statistics are extracted from each speech clip which are then mapped to the respective HIGH or LOW personality trait classification. The platform enables a user to choose from a variety of machine learning algorithms, feature selection methods and scoring values. This enables an evaluation of model performance on the aforementioned dataset.

The rest of this thesis is organized as follows: Chapter 2 discusses the Five Factor Model and the speech corpus used in this work. Chapter 3 discusses Machine learning and the algorithms that were used in this work. Chapter 4 discusses the feature extraction that was employed on the speech corpus. Chapter 5 discusses the framework's design, showcasing the graphical interface that facilitates classifier training and result visualization. Chapter 6 concludes this thesis and suggests future directions.

Chapter 2

The Five Factor Model and the Speaker Personality Corpus Dataset

In this study, we employ the Five Factor Model, commonly referred to as the Big Five personality traits or the OCEAN model. This chapter delves the model's comprehensiveness, addresses potential objections and their corresponding responses, and provides an explanation of the five traits of the model.

The five-factor model of personality [8] is a hierarchical organization of personality traits in terms of five basic dimensions: **Extroversion, Agreeableness, Conscientiousness, Neuroticism, and Openness**. Research using both natural language adjectives and theoretically based personality questionnaires supports the comprehensiveness of the model and its applicability across observers and cultures.

For decades factor analysts attempted to find the basic dimensions of personality by factoring personality scales. Tupes and Christal [9] found five recurrent factors in analyses of personality ratings and they were understandably surprised. Despite their work, not until the 1980s researchers from many different traditions were led to conclude that these factors were fundamental dimensions of personality.

2.1 Evidence of comprehensiveness

Amelang and Borkenau [10] collected both self-reports and peer ratings on a set of German adjective trait rating scales, and self-reports on a diverse set of personality inventories. Five factors were found in each data set which showed some similarities to the standard five. McCrae and Costa [11] used two data sources—self reports and peer ratings—and two instruments—adjective factors and questionnaire scales— to assess the five-factor model of personality. They showed convergence for all five factors across both observers and instruments. Similar findings have been reported by Goldberg [12], Ostendorf [13], and Trapnell and Wiggins [14].

2.2 Objections and responses

Many writers have argued that five factors are too few to summarize all that we know about individual differences in personality. As McCrae et al. [15] noted, measurement of the five factors gives a complete characterization of the person only at a global level. The factors represent groups of traits that covary, but are not necessarily interchangeable. More serious is the question of whether there are additional common factors not included among the Big Five. This is, of course, possible, though it appears increasingly unlikely, given the wealth of data in support of the comprehensiveness of the FFM.

On the other hand, some researchers argue that five factors are too many. Zuckerman et al. [16] argued that three factors, corresponding to H. J. Eysenck's E, N, and P are enough. The problem with all these proposals is that they are mutually inconsistent. It appears that all five factors are necessary, and this observation is supported by empirical analyses. McCrae and Costa [17] extracted factors from 80 adjective pairs in one sample of self-reports and one of peer ratings. They found that when they extracted fewer or more than five factors, those factors couldn't be consistently matched across two different samples (self-reports and peer ratings). However, with five factors, they observed a nearly perfect match between the two samples.

Hogan (in press) argues that the FFM is only suitable for describing the public self or social reputation (observer ratings), rather than reflecting inner drives and dispositions (self-reports). This objection is somewhat puzzling in view of the repeated recovery of the five factors in self-report data (Fiske [18] and McCrae & Costa [19]).

Another debate surrounding the Five-Factor Model is whether the FFM is a realistic description of human personality traits or merely a cognitive artifact resulting from how people perceive and judge others. People's

implicit personality theories, as revealed through their ratings of strangers and judgments of similarity in trait terms, appear to be structured by dimensions that closely resemble the FFM. This raises the possibility that the FFM is itself nothing more than a projection of our cognitive biases onto the targets we rate. A variety of ingenious studies have been devised to test this hypothesis, and although it still has some proponents, most personality psychologists have rejected it.

2.3 Description of the Five Traits

Openness: the tendency to be imaginative, independent, and interested in variety versus practical, conforming, and interested in routine.

Conscientiousness: the tendency to be organized, careful, and disciplined versus disorganized, careless, and impulsive.

Extroversion: the tendency to be sociable, fun-loving, and affectionate versus retiring, somber, and reserved.

Agreeableness: the tendency to be softhearted, trusting, and helpful versus ruthless, suspicious, and uncooperative.

Neuroticism: the tendency to be calm, secure, and self-satisfied versus anxious, insecure, and self-pitying.

Figure 2.1 illustrates the high and low ratings of the Big Five Traits.

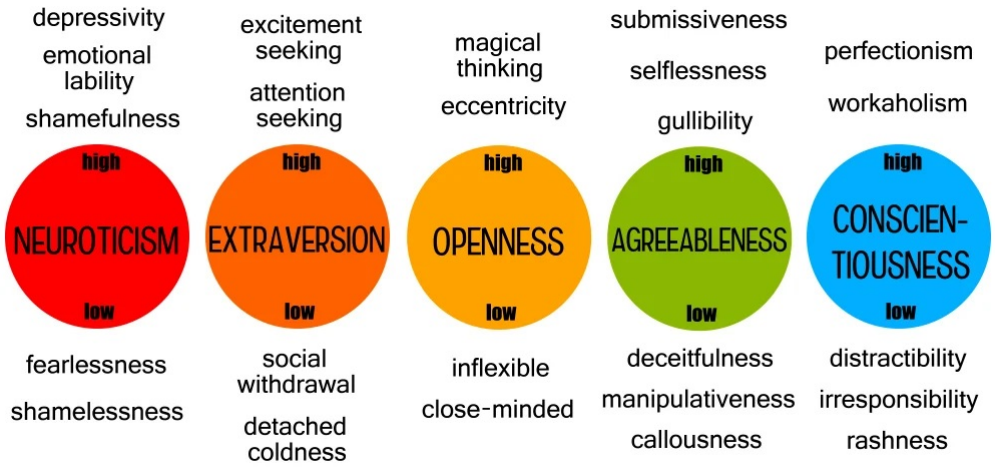


Figure 2.1: *High and low ratings of the Big Five Traits.*

2.4 Speaker Personality Corpus

The dataset used in this work is the *Speaker Personality Corpus* (SPC) which was used in the Personality Sub-Challenge. This Sub-Challenge is part of the INTERSPEECH 2012 Speaker Trait Challenge [20].

The SPC consists of 640 audio clips extracted from French news bulletins broadcasted by Radio Suisse Romande in February 2005. Each clip features one speaker, with a total of 322 individuals in the dataset. The majority of speakers appear only once (61%), while some appear twice (20.2%). The clips are around 10 seconds long, adding up to about 1 hour and 40 minutes of total audio. Figure 2.2 depicts the percentage of speakers appearing a given number of times. Roughly two thirds of the individuals represented in the corpus appear only once.

2.5 Personality Assessment

Each judge assigned scores for each of the Big Five personality traits for every clip. The BFI-10 questionnaire was completed through an online application. The judges worked a maximum of 60 minutes per day in two 30-minute sessions to maintain concentration. The order of clip presentation was randomized for each judge to prevent fatigue effects. The BFI-10 questionnaire is shown is Figure 2.3.

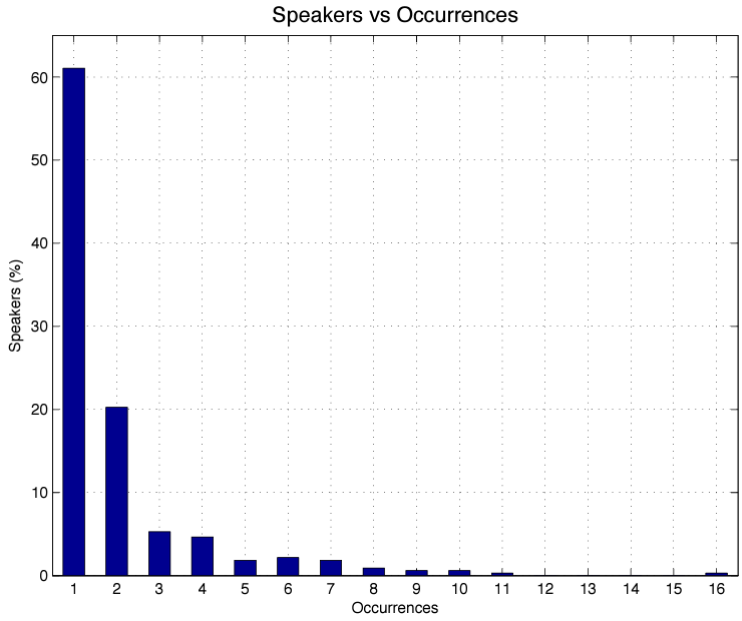


Figure 2.2: *Distribution of speaker occurences [from 7].*

ID	Question
1	This person is reserved
2	This person is generally trusting
3	This person tends to be lazy
4	This person is relaxed, handles stress well
5	This person has few artistic interests
6	This person is outgoing, sociable
7	This person tends to find fault with others
8	This person does a thorough job
9	This person gets nervous easily
10	This person has an active imagination

Figure 2.3: *BFI-10 questionnaire [from 7].*

Each question is associated to a 5-point Likert scale (from "Strongly Agree" to "Strongly disagree") mapped into the interval $[-2, 2]$. The BFI-10 includes the ten items that better correlate with the assessments obtained using the full BFI (44 items). The personality scores can be obtained using the answers provided by the assessors (Q_i is the answer to item i):

- Extraversion: $Q_6 - Q_1$
- Agreeableness: $Q_2 - Q_7$
- Conscientiousness: $Q_8 - Q_3$
- Neuroticism: $Q_9 - Q_4$
- Openness: $Q_{10} - Q_5$

2.5.1 Scoring

Each judge assigned scores for each of the Big Five personality traits for every clip. The BFI-10 questionnaire was completed through an online application. The judges worked a maximum of 60 minutes per day in two 30-minute sessions to maintain concentration. The order of clip presentation was randomized for each judge to prevent fatigue effects.

2.5.2 Language Constraints

The judges signed a declaration stating that they do not understand French, the language spoken in the clips. This was to ensure that only nonverbal cues were considered for personality assessment. Efforts were made to

exclude clips with potentially recognizable words for non-French speakers.

2.5.3 Labeling

Clips were labeled as "above average" (X) for a specific trait X (O, C, E, A, N) if at least six judges (the majority) assigned a score higher than their average for that trait. Otherwise, the clip was labeled as "NX."

2.5.4 Data Split

The dataset was split into training, development, and test sets using a speaker-independent stratification based on speaker gender.

Chapter 3

Machine Learning

Machine learning (ML) is a subset of artificial intelligence that involves the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed for each specific task. Tom Mitchell (1997) has defined ML as follows:

> A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Let's take the example of recommendation engines. The task in a recommendation system is to suggest content to users based on their preferences. The experience in a recommendation system consists of historical data that captures user interactions with items. The metric can assess how the user interacts with the recommended items, for example how many times the user clicked on the item (Click-Through Rate). The goal of the system is to optimize the evaluation metric.

Machine Learning is great for problems that require a lot of manual adjustments. For example in a spam filtering system we would have to write a lot of complex patterns and rules for words and phrases that we consider spam. With Machine Learning techniques the system automatically learns which words and phrases are spam. Moreover if spammers adapt to our filter, they can start using new rules. The Machine Learning system in this case can also adapt to their change, otherwise we would have to hard code new rules by ourselves. Another area where ML excels is for problems that are too complex for traditional algorithms. For example we might want to write a program that distinguishes between two specific words. We might notice that the first word starts with a high pitch sound, thus we could hardcode an algorithm that measures high-pitch sound intensity and use that to distinguish between the words. This obviously does not scale to thousands of words spoken by millions of people. The ML approach is to use an algorithm that learns by itself, given many example recordings for each word.

3.1 Types of Machine Learning Models

3.1.1 Supervised Learning

The algorithm is provided with a labeled dataset. Each data point is associated with a target value. The goal is to generalize this association in order to make predictions to new data.

3.1.2 Unsupervised Learning

The algorithm is provided with an unlabeled dataset. The goal is to find patterns, structure, or relationships between the data on its own.

3.1.3 Semi-Supervised Learning

The algorithm combines elements of both supervised and unsupervised learning. The labeled data helps guide the learning process, and the algorithm attempts to use the patterns it discovers in the labeled data to make predictions or inferences about the unlabeled data.

3.1.4 Reinforcement Learning

The learning system, called the *agent*, observes the enviroment, performs some actions and in return it gets *rewards* or *penalties*. The goal is to learn what is the best strategy so it can get the most reward over time.

3.2 Challenges

3.2.1 Data Quantity

Machine Learning algorithms require a substancial amount of data to learn effectively. Michele Banko and Eric Brill [21] evaluate the performance of different learning methods on an NLP task, when trained on orders of magnitude more labeled data than has previously been used. They proposed that "a logical next step for the research community would be to direct efforts towards increasing the size of annotated training collections, while deemphasizing the focus on comparing different learning techniques trained only on small training corpora."

3.2.2 Sampling Bias

A pitfall of data collection is the occurrence of sampling bias. Imagine a study that wants to understand the popularity of a particular movie by collecting data through social media, assuming that those platforms adequately represent the general view about the movie. However this intorduces sampling bias, because people who don't use these platforms are excluded from the study leading to a a skewed representation of public view.

3.2.3 Overfitting

Imagine going to a restaurant in a foreigh country and the food is bad. You might say that *all* food in this country is bad. We often make generalizations that are too wide, and machines can do the same if we're not careful. In Machine Learning this is called overfitting: it means that the model performs well on the training data, but it does not generalize well. Complex models can detect subtle patterns in the data, but if the training set is noisy then the model is likely to detect patterns in the noise itself. These patterns will not generalize to new instances.

3.3 Testing and Validating

The standard method to find out how well your model works on data it has never seen before it to split the given data into two sets: *training set* and *testing set*. The algorithm learns from the training set and uses the testing set to measure its performance. By evaluating the model on the test set, we get an estimate on how the algorithm will perform on cases it has never seen before. If the training error is low but the testing error is high then this means the model is overfitting.

3.3.1 Hyperparameter Tuning

Now suppose we need to decide between two models, for example a linear one and a polynomial one. One option is to train both and compare how well they perform on the testing set.

Now, imagine the linear model actually performs better in generalization, and we're considering using some regularization to prevent overfitting. The issue is, how do we pick the right value for the regularization hyperparameter? We could train a total of 100 models, each with a distinct value for this hyperparameter. Suppose we manage to identify the optimal hyperparameter value that leads to a model with only a 5% error—initially, this appears to be a promising achievement. However, once we implement this chosen model, it unexpectedly produces errors of 15%, which is disappointing. The problem is that we measured the error multiple times on the same test set, thus adapting the model and the hyperparameters to produce the best result for that particular set.

A common solution to this problem is to simply hold out part of the training set to evaluate several candidate models and select the best one. That set that he hold out is called *validation set*. We train multiple models with various hyperparameters on the reduced training set (the full training set minus the validation set), and we select the model that performs best on the validation set.

3.4 Feature Engineering

Feature engineering is a critical step in the process of machine learning. It involves transforming raw input data into a format that is suitable for use by machine learning models. Well-engineered features can significantly impact the performance and effectiveness of a machine learning model.

Features are characteristics of a dataset that are used as input variables for a machine learning process. Features provide information about the data that can be used by machine learning algorithms to make predictions or classifications.

Feature engineering consists the following elements:

- feature construction
- feature extraction
- feature selection

3.4.1 Feature Construction

Feature construction involves creating new features from the existing ones to capture more meaningful and informative patterns in the data. Let's consider an example of feature construction in the context of a customer transaction dataset. The dataset might contain the following features:

CustomerID, Transaction Date, Product Id, Transaction Amount

From these features, we could constuct a new feature, namely

Month of Purchase.

This feature could help identify seasonal trends in customer spending.

3.4.2 Feature Extraction

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. There are two main approaches to feature extraction, manual and automated:

Manual: This approach involves human intervention to select and define relevant features. Domain experts manually design and compute these features based on their understanding of the problem. For instance, in natural language processing (NLP), manual feature extraction might involve creating features like word frequency counts, TF-IDF scores etc.

Automated: This approach employs algorithms or techniques to automatically derive relevant features from the raw data. Automated feature extraction is particularly valuable when dealing with high-dimensional or complex data, such as images, audio, or sensor data.

3.4.3 Feature Selection

Feature selection consists of selecting a subset of features which make the most meaningful contrubution in a machine learning activity. Feature selection reduces the dimensionality of the dataset by eliminating redundant features. This leads to a reduction in the computational resources required for training the machine learning model.Removing irrelevant or noisy features through feature selection can improve the efficiency of the learning model. Irrelevant features can introduce noise into the model and may lead to overfitting. Feature selection can lead to a more interpretable model. When we work with a reduced set of selected features, it becomes easier to understand and explain the factors that influence the model's predictions.

3.5 Feature Selection Methods

This section outlines the feature selection methods used for this study. The methods that are used are the following:

- recursive feature elimination (RFE)
- select K best
- principal component analysis (PCA)

3.5.1 Recursive Feature Elimination

RFE requires an external estimator, which is typically a machine learning model that assigns weights or importance scores to each feature (e.g., the coefficients of a linear model). It iteratively selects features in multiple rounds. In each round, the external estimator is trained on the current set of features, and feature importance scores are calculated based on the estimator's output. RFE identifies and removes the least important features from the current set. The pruning step is recursively repeated on the produced set until the desired number of features to select is eventually reached.

3.5.2 Select K Best

This method selects features according to the k highest scores that are produced by a scoring function. In this work we are using **f_regression** from the **Scikit-Learn** library. It is used to compute the F-value and p-value for each feature in a dataset concerning the target variable.

3.5.3 Principal Component Analysis

PCA is not exactly a feature selection method. In feature selection we select and exclude features without changing them. Dimensionality reduction transforms features into a lower dimension.

PCA is a statistical technique to transform a set of correlated variables into a set of uncorrelated variables called principal components. The principal components are vectors and are a linear combination of the original variables. They are orthogonal to each other (uncorrelated). Since principal components are uncorrelated, they capture the maximum amount of variability in the data. The first principal component is computed so that it explains the greatest amount of variance in the original features. The second component is orthogonal to the first, and it explains the greatest amount of variance left after the first principal component and so on.

3.6 Machine Learning Algorithms

This section outlines the machine learning algorithms used for this study. The methods that are used are the following:

3.6.1 Support Vector Machines

SVM is a method that can be used in both classification and regression. The core idea of SVM is to determine a *hyperplane* which draws a boundary between data instances plotted in the multi-dimensional feature space. A hyperplane for an N dimensional feature space, is a flat subspace of dimension $N - 1$ that separates and classifies the dataset.

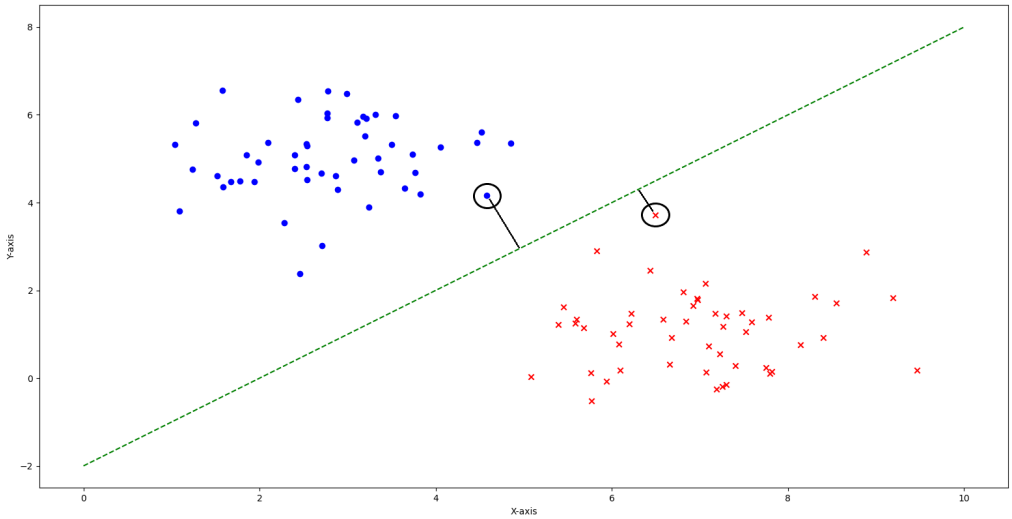


Figure 3.1: *1-D hyperplane separating data points in 2-D.*

Therefore, the goal of SVM is to find a hyperplane that best separates the data points of different classes. The hyperplane is chosen so that it maximizes the distance between the hyperplane and the nearest data points from each class. This distance is called the *margin*. As shown in Figure 3.1, the green line represents a 1-D hyperplane separating the data points into two classes in a 2-D space. The points that are circled are the ones closest to the hyperplane and are called *support vectors*. If are removed, they will alter the position of the dividing hyperplane.

The data may not be linearly separable like the example above, meaning that a single hyperplane cannot cleanly separate the different classes. SVM uses *kernel functions* to transform the input data from a lower-dimensional space into a higher-dimensional space, where it might become linearly separable, as in Figure 3.2.

3.6.2 Logistic Regression

Logistic Regression is a method used to model how the odds of a binary response depend on a set of predictors. The method is useful when we want to understand the relationship between predictor variables and an event

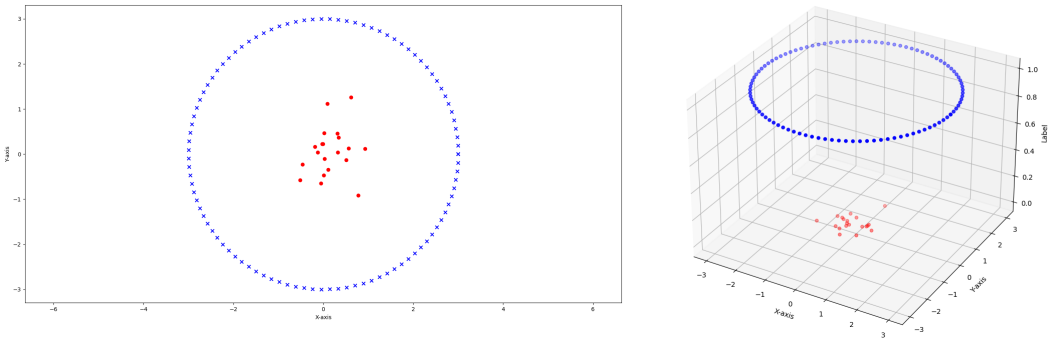


Figure 3.2: *Non linearly separable data in 2D, transformed to linearly separable in 3D.*

occuring. Logistic Regression belongs to the class of Generalized Linear Models (GLM). A GLM consists of three parts:

- **random componen**t: Specifies the probability distribution of the response variable. The choice of the distribution depends on the nature of the response variable. For example,in linear regression, the random component assumes a normal distribution. In binary logistic regression, it assumes a binomial distribution.
- **Systematic Component:** Specifies the relationship between the response variable and the predictors in the model. It defines a linear combination between them.
- **Link Function:** Specifies the link between the random and the systematic components. The link function transforms the linear predictor (from the systematic component) into the expected value of the response variable.

In Logistic Regression, we assume a binomial distribution for the response variable, with **p** representing the event's probability. The systematic component combines predictors and parameters linearly. The Link Function is the **logit**, transforming this combination into log-odds of success. Mathematically it is represented as:

$$logit(p) = \log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k = x^T \beta$$

Solving for p we get:

$$p = \frac{e^{x^T \beta}}{1 + e^{x^T \beta}}$$

This nonlinear function is a sigmoidal function, depicted in Figure 3.3, of the model terms and constrains the probability estimates to between 0 and 1.

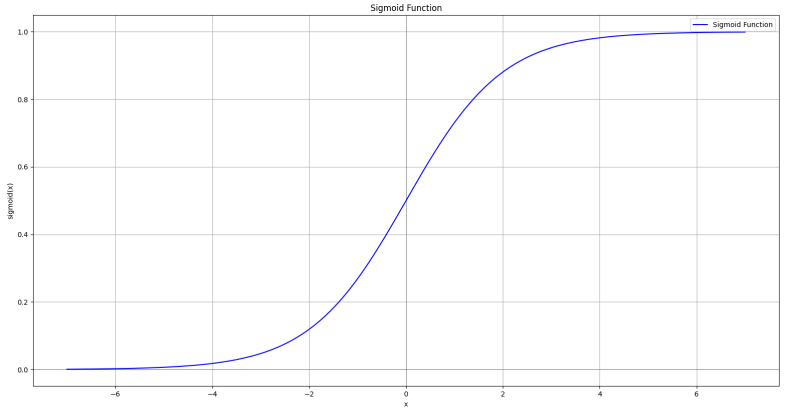


Figure 3.3: *Sigmoid function.*

After relating the model to the parameter of the binomial distribution we can find candidate values of the parameters β that maximize the log likelihood of the observed data.

3.6.3 Gaussian Naive Bayes

Naive Bayes is a probability based classifier. To classify a new data point, the classifier calculates the posterior probabilities for each class and selects the class with the highest probability as the predicted class. Specifically, for a test example $\mathbf{x}$ it reports the class y that has the highest value of $p(y|\mathbf{x})$. If the largest value is achieved by more than one class, it chooses randomly from that set of classes.

The problem is that we usually do not have $p(y|\mathbf{x})$. However, we could use Bayes rule as follows:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y) \cdot p(y)}{p(\mathbf{x})}$$

Now the problem lies with the probability $p(\mathbf{x}|y)$. To address this problem, the classifier assumes that the features are conditionally independent given the class:

$$p(\mathbf{x}|y) = \prod_j p(\mathbf{x}^j|y)$$

Thus:

$$p(y|\mathbf{x}) \propto (\prod_j p(\mathbf{x}^j|y)) \cdot p(y)$$

So the rule that the classifier follows in the case where all the errors have the same cost is:

$$\text{choose y such that } [(\textstyle\prod_j p(\mathbf{x}^j|y)) \cdot p(y)] \text{ is largest}$$

Lastly, because multiplying a large number of probabilities can lead to numerical underflow it is practical to work with log probabilities:

$$\text{choose y such that } [(\textstyle\sum_j \log(\mathbf{x}^j|y) + \log p(y)] \text{ is largest.}$$

The *Gaussian Naive Bayes* is a specific variant of the Naive Bayes algorithm that assumes continuous features follow a Gaussian (normal) distribution. In other words, it assumes that the numerical features in your dataset are normally distributed within each class.

We can use other parametric models for $p(\mathbf{x}^j|y)$, depending on the nature of the $\mathbf{x}^j$. Bayesian classifiers use all the available information to make predictions, even if some parts of that information have only a small impact. Many other methods tend to ignore the less important details. Bayesian classifiers believe that even if a few individual pieces of information don't matter much on their own, when you put them all together, they can make a big difference in making predictions. This makes Naive Bayes especially good at certain types of tasks where this idea is helpful.

3.6.4 Decision Trees

The Decision tree algorithm builds a model in the form of a tree structure. Each internal node is labeled with an input feature. The edges from each internal node are labeled with every possible value that the feature represented by the node can take. The leafs contain class labels. In the process of classifying, the test example is compared against the root node. The edge that corresponds to the same feature values on the test example and the root node is followed. The process continues until a leaf node is met. The class of the leaf is then returned.

In Figure 3.4 we give an example of a decision tree built from a test dataset. The datasets consists of two features. Weather can take three values and Wind can take two. The output variable is weather someone will want to go out based on those conditions.

Example	Weather	Wind	Class
ex1	Sunny	Strong	Yes
ex2	Cloudy	Weak	Yes
ex3	Rainy	Strong	No
ex4	Sunny	Weak	No
ex5	Cloudy	Strong	Yes
ex6	Rainy	Weak	Yes

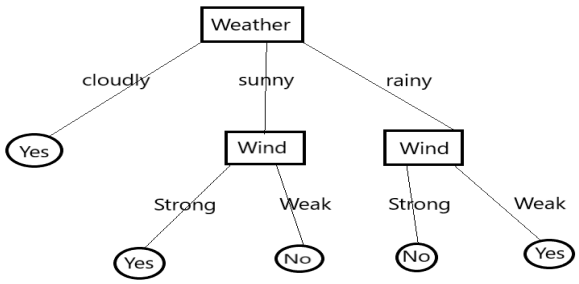


Figure 3.4: *Test data and the corresponding decision tree.*

Decision trees are created using a method called recursibe partitioning. The method divides the training data into smaller groups based on their feature values. The process starts at the root node that represents the entire dataset. The algorithm identifies the feature that has the strongest influence on predicting the target class. The decision tree then splits the data into various partitions, each characterized by a unique value of the selected feature. These partitions become the initial branches of the tree. This continues until certain stopping conditions are met, usually when the tree has grown to a predifined limit, all the available features have been used, or most of the examples in a node belong to the same class.

The primary challenge of a decision tree is to determine which feature to select to split the data. The goal is to split the data in a way such that the partitions created belong to a single class. The partitions in that case are said to be pure. A measure of impurity is Entropy and is used by many algorithms (ID3, C5.0). Information gain is then computed based on the reduction in Entropy after the dataset is split based on a specific feature. Therefore, constructing decesion trees is about identifying the attribute that yields the highest information gain.

Generally, smaller trees are preferred. A human expert can interpret the results better when the decision tree that consists of no more than a few tests. Also, smaller trees tend to dispose irrelevant and redundant information.

3.6.5 k-Nearest Neighbor

The k-Nearest Neighbor (kNN) is a classification algorithm. The algorithm assigns a class label to the unknown data point based on the class labels of the most similar training data points.

There are many measures of similarity. The most common one is Euclidean distance. For $\mathbf{x} = (x_1, x_2, ..., x_n)$, $\mathbf{y} = (y_1, y_2, ..., y_n)$ the Euclidean distance is:

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

The algorithm calculates the distance between the test example and each of the training examples. It takes the majority class of the **k** closest examples to **x** as the class label of **x**. If there are ties it is left to us to decide a mechanism that decides a class.

Selecting the right value of **k** is a crucial decision and can significantly impact the performance of the kNN classifier. If the value of **k** is large the algorithm essentially considers all data points as neighbors. It may then become too biased by the majority class, and it might fail to capture local patterns or variations in the data. This can result in poor generalization to new, unseen data. Conversely, when **k** is set to a very small value, the classifier becomes highly sensitive to noise or outliers in the training data. The best value for **k** typically lies somewhere between these two extremes.

Because the algorithm is based on distance it is affected by the scale of the variables. For example we might have in a dataset one variable of "house size" and one for "house price", the first being in the thousands while the second on the hundred of thousands. If we apply the kNN algorithm directly to this dataset without scaling, it could lead to biased results because the "house price" feature would dominate the distance calculations due to its larger magnitude. The primary goal of feature scaling is to ensure that all features contribute equally to the modeling process and that no single feature dominates due to its scale or magnitude. We could use min-max scaling to scale both features to a common range of [0, 1]. Then the kNN algorithm can work effectively without any single feature dominating the distance calculations.

kNN is commonly referred as a "lazy learner," indicating that the algorithm refrains from making generalizations based on the training data points. In simpler terms, KNN lacks a distinct training phase and retains the entire training dataset without making assumptions about the underlying data, making it a non-parametric learning approach.

Nevertheless, kNN is a simple algorithm and easy to understand. It is effective in certain situations, e.g. for recommendation systems. It is also very fast, requiring very little or almost no time to prepare for making predictions.

3.6.6 Random Forest

Random Forest is an ensemble classifier, i.e. it combines multiple models to improve predictive performance. Random Forest combines multiple decision trees to achieve this. The decision trees are built using bagging. Bagging is a technique that involves creating multiple subsets (bags) of the original dataset by randomly selecting data points with replacement. Each subset is used to train a separate decision tree. After the random forest is generated by combining the trees, majority vote is applied to combine the output of the different trees.

The out-of-bag (OOB) error rate is used to assess the performance of a Random Forest model. The samples left out of the subset sample dataset and not used in the construction of the i-th tree can be used to measure the performance of the model. For example, consider the i-th decision tree T_i that has been fitted on a subset of

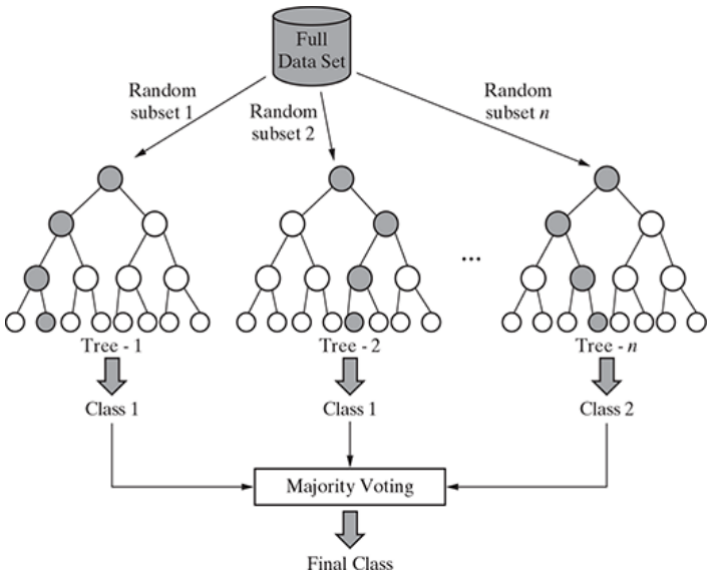


Figure 3.5: *Random Forest Classifier [22].*

the sample dataset. For every training example (x_i, y_i) not in the sample subset of T_i, we can use T_i to predict the output o_i for x_i. Therefore, the error $|o_i - y_i|$ can be easily computed. Thus, the out-of-bag error is the average value of this error across all decision trees.

Random forests reduce the risk of overfitting by aggregating the predictions of multiple decision trees. Each tree in the forest is trained on a different subset of the data and potentially with a different subset of features. This diversity helps reduce overfitting because individual trees may capture noise in the data, but when averaged, they tend to produce a more generalized prediction. Random forests are also robust to missing data, due to bagging. This means they can maintain their accuracy even when a portion of the data has missing values, making them valuable in cases where data may not always be complete.

Random forests are more challenging to interpret compared to a single decision tree. A random forest is an ensemble of many individual decision trees, thus Understanding how each tree contributes to the final prediction is harder. In contrast, a single decision tree is typically easier to interpret and visualize. Random forests are also more computationally more expensive than a simple decision tree model. This is because random forests involve training and evaluating multiple decision trees in parallel.

3.6.7 Boosting

Boosting is a powerful ensemble learning technique that combines multiple weak learners (typically simple models) to create a strong predictor. The process of boosting involves constructing a new weak learner in a way that focuses on the mistakes made by the previous ones. This helps to incrementally improve the overall prediction accuracy.

AdaBoost
AdaBoost fits a sequence of weak learners on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The algorithm initially applies weights to each traning example. At the beggning, each weight is $w_i = 1/N$, where N is the number of the training examples. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. The training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. Thus, the weak learners in subsequent iterations are forced to pay more attention to difficult examples and try to classify them correctly.

XGBoost
Extreme Gradient Boosting (XGBoost) is an implementation of the gradient boosting algorithm. Gradient Boosting is an ensemble learning method that combines the predictions of multiple individual models to create a strong predictive model. The key concept behind gradient boosting is to sequentially build models that focus on reducing the errors made by the previous models. This is achieved through the process of gradient descent, where the objective is to minimize a loss function. The choice of the loss function depends on the type of problem you're dealing with. For binary classification, it's often the log-likelihood loss or cross-entropy loss. Gradient descent is used to find the direction and magnitude of the steepest decrease in the loss

function's value. The gradient of the loss with respect to the model's predictions guides how the model should be adjusted to reduce the error. In each iteration of gradient boosting, a new model is trained on the residuals (the differences between the actual target values and the predictions of the current ensemble). The goal of this new model is to capture the patterns in the residuals that the previous models couldn't. This process is repeated for a specified number of iterations or until a certain level of accuracy is reached.

3.7 Performance Metrics

This sections outlines the performance metrics that are used in this study. We are using confusion matrices and ROC curves.

3.7.1 Confusion Matrix

When evaluating the performance of a binary classifier, the error rate alone may not provide enough information about the classifier's performance. A confusion matrix is a technique for summarizing the performance of a classification algorithm. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It provides insights into not just the errors made by your classifier, but also the specific categories of errors that are occurring. The steps involved in calculating a confusion matrix are:

- Begin with a test dataset or a validation dataset that includes the expected outcome values for each data point.
- Generate predictions for each individual data point in your test dataset using your classification model.
- Based on the expected outcomes and the model's predictions, count the number of correct predictions for each class and the number of incorrect predictions for each class, organized by the class that was predicted.

Finally, based on these counts, each row of the matrix corresponds to a predicted class and each column of the matrix corresponds to an actual class. The counts of correct and incorrect classification are then filled into the table.

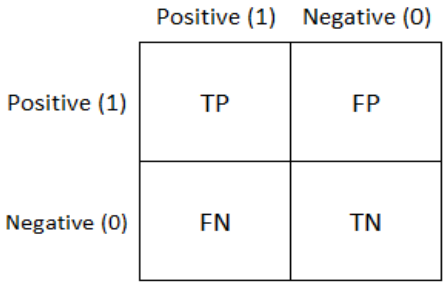


Figure 3.6: *An example of a Confusion Matrix.*

- **TP**: number of instances that were correctly predicted as positive.
- **FP**: number of instances that were incorrectly predicted as positive.
- **FN**: number of instances that were incorrectly predicted as negative.
- **TN**: number of instances that were correctly predicted as negative.

3.7.2 ROC Curve

Another evaluation metric for any classification model is the AUC-ROC curve. The ROC (Receiver Operator Characteristic) curve is a graphical representation of a binary classification model's performance. It helps assess how well a model can distinguish between two classes (typically positive and negative) by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR).

$$\textbf{TPR} = \frac{TP}{TP + FN}$$

$$\textbf{FPR} = \frac{FP}{FP + TN}$$

The Area Under the Curve (AUC) is the measure of the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve. A perfect classifier would have an AUC of 1.0, while a random classifier would have an AUC of 0.5. Generally, the higher the AUC, the better the model's ability to discriminate between positive and negative cases.

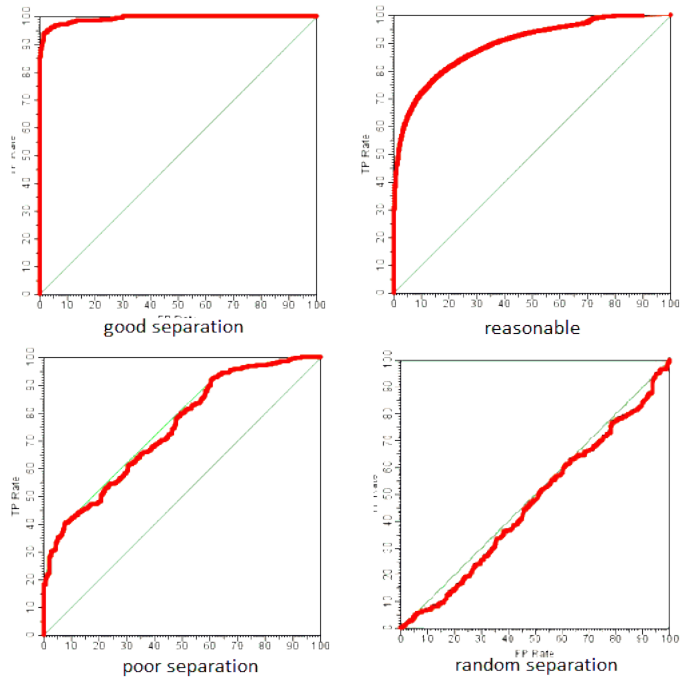


Figure 3.7: *Receiver Operating Curve (ROC) Analysis [23]*

Chapter 4

Feature Extraction

4.1 Introduction

According to the American Psychological Association, "**Personality** refers to individual differences in characteristic patterns of thinking, feeling and behaving". Individual differences have been subjected to scientific inquiry since the times of Theophrastus, the Greek philosopher, who posed the question: "Why is it that, while all Greece lies under the same sky and all the Greeks are educated alike, it has befallen us to have characters so variously constituted".

Capturing crucial aspects of a person's individuality is a task receiving increased interest in the computing community and this trend does not seem to go downwards in the near future since personality is relevant to any computing area involving understanding, prediction, and synthesis of human behavior.

Measuring personality is not an easy task to be performed even by humans. Different theories have been adopted as a personality basis but the most widely recognized is the so-called trait theory. Personality traits are factors capable of capturing stable individual characteristics underlying overt behavior. These traits are known as Big Five (Big-5) or Five-Factor Model (FFM) [24] and they are also widely accepted in the computing community: Extraversion, Agreeableness, Conscientiousness, Neuroticism, Openness.

In Automatic Personality Perception, one tries to infer the personality an observer has attributed to a person using a computer program. Such a program requires the actual personality scores of an individual as provided by a rater utilizing some personality representation model such as the Big-5 Inventory 10 (BFI-10). BFI-10 is a very short and common questionnaire for personality assessment that correlates very well to the original Big-5 Inventory. In BFI-10, each question is associated with a 5-point Likert scale (strongly disagree to strongly agree, respectively) mapped into the interval [-2,2]. The personality scores can be obtained using the answers provided by the raters, averaged by the number of raters.

Thus, in the computing literature, there are quite a few works connecting personality prediction and data such as text, audio, and video. Only works based on audio will be discussed here.

4.2 Features for Automatic Personality Perception from Speech

Psychologists suggest that verbal communication is capable to externalize the personality of an individual and to influence the traits that others attribute to it: from a computational point of view, this means that the personality of an individual can be inferred to some extent from automatically detected verbal behavioral cues. Moreover, it has been observed that judgements made from speech alone consistently have the highest correlation with the whole person's judgements [25]. Here, the term "speech" also carries paralinguistic phenomena such as prosody, vocalization, emotions, etc. Although there have been recent competitions and challenges in speech-related conferences, there seems to be no specific feature set or machine learning (ML) approach that clearly outperforms others [26].

In what follows, we will shortly describe works that connect speech features, ML approaches and personality traits. The main focus of this review is on speech features.

4.2.1 Related work

In [27], low level descriptors (LLDs) are extracted, statistics are calculated, and mapping into trait attributes is performed. The LLDs are **pitch, formants, energy, and speaking rate (as length of voiced and unvoiced segments)**. A 40 ms analysis window and a 10 ms frame rate is suggested, while PRAAT is used for the extraction. Four statistics are suggested on the LLDs: **average, minimum, maximum, relative entropy**. In total, 24 features are extracted for each audio clip. The SSPNet corpus is used. For assessment,

10-s long clips (spontaneous trait inference) were randomly extracted and given to the raters. The features where extracted from the same clip. For classification, the scores of each trait are split in two subsets, High (samples that have a score higher than the average) and Low (samples that have a score lower than the average). Each set is a class and an SVM-RBF is used for the classification in a k-fold cross validation (CV) framework, with k=15. Also, for each dimension, the clips have been split into the same groups, High and Low.

In [28], [29], LLDs are extracted every 10 ms, statistics are then obtained at the utterance level, and thus 1450 features are generated. PRAAT is used for extraction and the LLDs are **intensity (dB), pitch, loudness, formants (PLP-based + bandwidths), spectrals (centroid, 95% roll-off point of spectral energy, flux), MFCCs (16)**, and **other (HNR, ZCR, rhythm)** features. As for statistics, **means, 1-4 moments, extrema, and ranges** are computed. Signal is also divided into **voiced, unvoiced, and silence** parts. Ranking of features by information gain (IGR) is performed. MFCCs predominate the set. SVMs with linear kernels have been used for classification (pair-wise), with 10-fold CV. In [30], a similar work to [27] is presented. It is suggested that prosodic features can be reliably extracted from syllabic nuclei, which are the speech segments that most likely correspond to vowels. Syllables are identified as speech segments enclosed between consecutive energy minima and the nuclei is identified as regions that lie within -5 dB from the energy peak that characterizes each syllable. Intonation features comprise of energy, pitch, and syllable length related measures. A 40-ms analysis window is applied with a 10-ms frame rate. The average of the multiple values extracted from the same nucleus are used as representative value of the syllable. Voice quality features consist of harmonicity measures, spectral centroid, spectral skewness, spectral kurtosis, spectral slope. The set is completed using jitter, shimmer, and glissando likelihood. Each nucleus is represented by a vector of these short-term features. For statistics, the mean of all features is computed, the standard deviation for nuclei and syllable length is computed, as well as for pitch, energy, spectral slope, harmonicity, and spectral centroid. The entropy is also computed in these features. Mean and bandwidth of the first three formants are also extracted from each nucleus. Finally, minimum of pitch and maximum of energy completes the feature set, a total of 35 features. Ordinal regression is used for ranking people according to the personality traits attributed by assessors.

In [31], the authors of [27] suggest a similar approach, extracting LLDs such as pitch, the first two formants, the energy of the signal, and the length of voiced and unvoiced segments using PRAAT. Again, the analysis window is of 40 ms and the rate is 10 ms, making a final vector for each frame consisting of six features which are transformed to z-scores (using mean and standard deviation estimated in the training set). For statistics, minimum, maximum, mean, and relative entropy of the differences between LLDs extracted from consecutive analysis windows are considered. The feature vector consists of 24 elements for each speech clip. Logistic regression and an SVM are used for classifying feature vector in one of the two classes associated with each personality trait (Low, High) in a k-fold CV framework (speaker-independent) on the SSPNet dataset. In the work presented in [32], the authors propose the use of modulation spectrum to recognize traits. Due to the high dimensionality of the feature space, the Kolmogorov-Smirnov statistical test is used to select useful features. The feature set includes a baseline feature set (6125 values per speech clip) provided by the organisers of a challenge. Adaboost is used for classification. In [33], the authors use an agent-caller paradigm to collect data and a very large set of features is automatically extracted via openSMILE. Features include pitch, MFCCs, spectral parameters, as well as a great variety of statistics (range, centroid, standard deviation, skewness, kurtosis, min-max values, quartiles and percentiles etc.). A total of 6552 features were extracted per speech clip. The type of classifier is not mentioned.

In [34], the authors use the openSMILE feature set with a k-fold speaker independent training on SVMs for each trait but they extend it using pitch and intonation features, along with an amount of functionals. They also make use of the openSMILE toolkit for baseline feature extraction. The MOMEL/INTSINT intonation modeling software is used. Spectral features are also suggested. In [35], a Fast Sequential Floating Forward Search algorithm is presented to select features appropriate for classifying traits. The authors select a set of "families" that accounts for 129 members, including RASTA-style features, harmonic-to-noise ratio, pitch, jitter, MFCCs (1-14), shimmer, ZCR, probability of voicing, energy, etc. An SVM is suggested for a k-fold CV scheme. For all traits, MFCCs and RASTA-style spectrum features are always among the five first five families. In [36], the authors propose two prosodic and two spectral sets of features, in addition to the 6152 features from openSMILE. Prosodic features include pitch, normalized (by the overall speaker mean) pitch, z-scored pitch, voiced energy (RMS divided by maximum energy), and spectral tilt (all in log domain). Statistics are computed, max, min, range, mean, standard deviation over the whole utterance. Additionally, the authors fit a 5-order polynomial to obtain coefficients describing the dynamics of pitch and energy features. Moreover, MFCCs are used, along with the first three derivatives, as well as shifted delta cepstrum (SDC). Many classifiers were proposed, with SVM, GMM, Eigenchannel, among others.

In [37], a multiresolution analysis using the Wavelet Transform is proposed on top of the openSMILE feautures. Wavelet coefficients are used for the classification (approximation and detail). K-fold CV was used on SSPNet dataset with a set of neural classifiers, one for each trait. The authors of [38] propose a multimodal (lexical and acoustic) deep learning-based model. For the audio part, 384 LLDs such as pitch, intensity,

spectral and cepstral representations, duration, voice quality (jitter, shimmer) and HNR, spectral harmonicity, and psychoacoustic spectral sharpness were used. New datasets are evaluated (myPersonality, CXD corpus). Multi-layer Perceptrons and LSTMs were used for the classification. In [39], HMMs along with SVMs were suggested to classify traits from speech. Energy, MFFCs, and pitch information are extracted. The SSPNet corpus is used.

The authors in [40] propose the extraction of MFCCs along with 1st and 2nd derivatives feeding a CNN with a Gram Matrix layer for speaking style estimation. The First Impressions dataset is used consisting of 10000 audio clips extracted from videos.

4.2.2 Proposed Feature Set

Conclusively, the literature review suggests many different feature sets and much more classifying schemes for personality trait assessment from speech signals. To avoid the dimensionality curse, we would like to focus on features that are low in number but important in trait prediction. There seems to be an agreement that prosodic and spectral features (jointly termed as acoustic features) combined are able to predict personality traits to a certain level.

For our task we can choose a minimal (baseline) set of LLDs consisting of:

- Energy
- Pitch
- First and second formant values
- First and second formant bandwidths
- Zero Crossing Rate

and a more complex (enriched) set of LLDs that includes the baseline features plus

- Intensity
- Spectral measures (flux, entropy, 90% rolloff, centroid, spread)
- MFCCs
- First derivatives of all features (the so-called "deltas")

A set of statistics will be applied on these LLDs. Let $v = \{x_1, x_2, x_3, ..., x_N\}$ denote a sample LLD consisting of N values.

mean: The average value of an LLD

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

standard deviation: Measures the amount of variation or dispersion of an LLD

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})}$$

maximum: The maximum value of an LLD

$$M = \max\{x_i\}$$

minimum: The minimum value of an LLD

$$m = \min\{x_i\}$$

skewness: measures of the asymmetry of the sample distribution of an LLD about its mean

$$b_1 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^3}{[\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2]^{3/2}}$$

kurtosis: measures of the "tailedness" of the sample distribution of an LLD

$$g_2 = \frac{(N+1)N}{(N-1)(N-2)(N-3)} \frac{\sum_{i=1}^{N}(x_i - \overline{x})^4}{[\sum_{i=1}^{N}(x_i - \overline{x})^2]^2} - 3\frac{(N-1)^2}{(N-2)(N-3)}$$

and these will be the final features for each trait.

Reasons for such a selection are (a) ease of implementation, either directly or by opensource software, (b) robustness, (c) relevance to the task, (d) suitable for near-real time analysis.

4.3 Technical details

Speech is a non-stationary signal, that is, it quickly changes over time. Thus we need to process it in short segments (the so-called frames) that are approximately stationary. Stationarity implies that temporal and spectral characteristics inside a speech frame do not significantly change. The length of each frame should be long enough to reliably estimate a feature set but short enough to ensure the measured features are representative of the whole speech frame. For these reasons, the usual frame length is 20-40 ms. For example, a 30-ms window slides on the speech waveform with a step of 15 ms (the so-called step size or frame rate). This way, we have a 50% overlap between successive frames. Finally, each frame is windowed (multiplied by a function in time that has desirable properties in the spectral domain) by a Hamming window.

We will now briefly discuss the selected acoustic features and their properties that make them suitable for our task.

4.3.1 Energy

The energy of the m-th speech frame is defined as

$$E[m] = \frac{1}{2M+1}\sum_{n=-\infty}^{\infty} x^2[n]w^2[m-n]$$

where $x^2[n]$ is the speech frame squared and $w^[n]$ is the chosen window function squared with support in [n-M, n+M]. An example of energy contour variation (in dB) is shown in Figure 4.1. We can see that high energy values corresponds to highly sonorant parts of speech while low energy values are present in voiceless parts of speech.

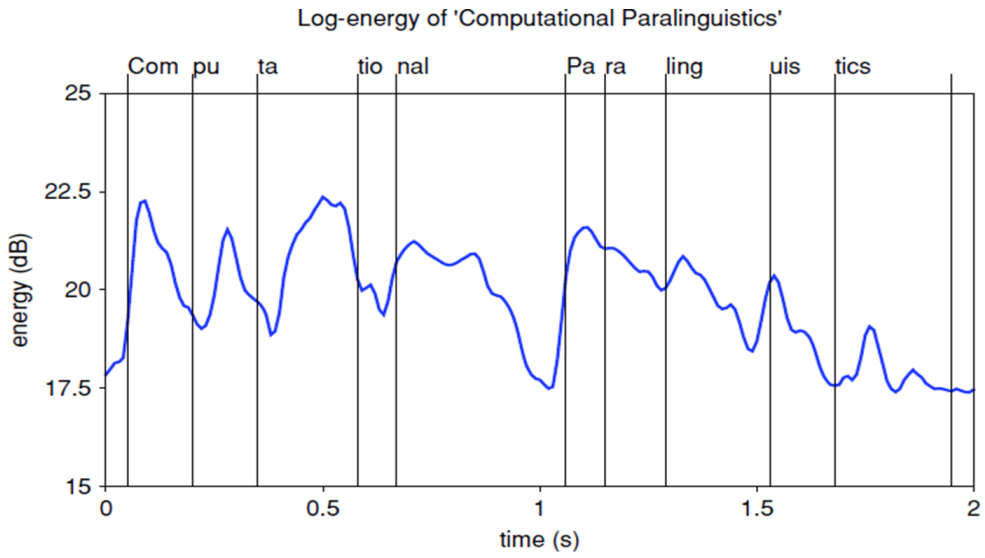


Figure 4.1: *Energy contour of a speech waveform.*

4.3.2 Fundamental frequency

It is well known that human perception is far more sensitive to changes in the fundamental frequency (or F0) than any other speech parameter. Thus, F0 has a significant influence on the performance of algorithms for emotion detection in speech and is considered as one of the most difficult tasks in speech processing. We propose to use the Probabilistic YIN (PYIN) algorithm, a modification of the well-known YIN algorithm for fundamental frequency (F0) estimation. PYIN jointly considers multiple pitch candidates based on a probabilistic interpretation of YIN. A prior distribution on the YIN threshold parameter yields a set of pitch candidates with associated probabilities, computed using YIN. The procedure effectively turns the popular frame-wise YIN algorithm into a probabilistic machine which outputs pitch candidates with associated probabilities. Temporal smoothness is obtained by using the candidate probabilities as observations in an HMM, which is Viterbi-decoded to produce the final pitch track. An example of the fundamental frequency contour is shown in Figure 4.2.

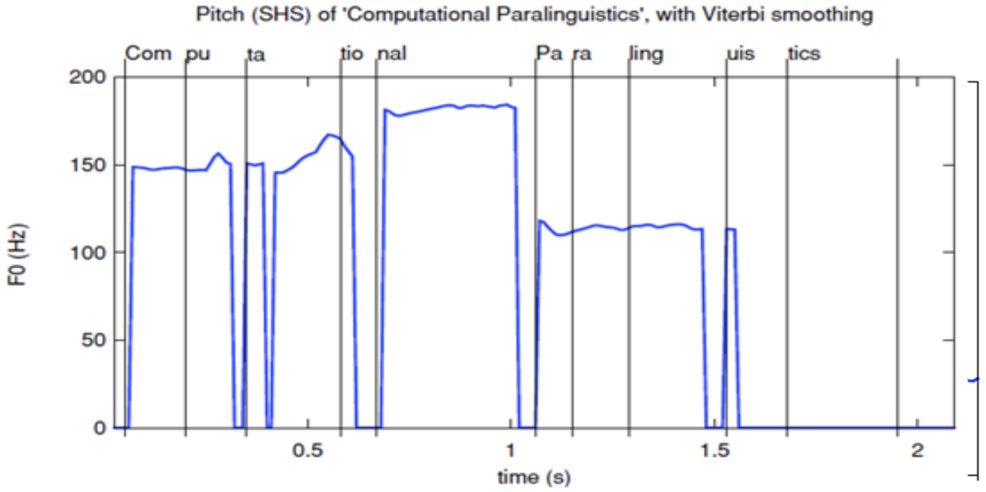


Figure 4.2: *F0 contour of a speech waveform.*

4.3.3 Zero Crossings Rate

The ZCR provides some rough information about the frequency distribution of the speech signal and is defined as the number of times a signal crosses the horizontal (time) axis: a high frequency signal should have a high number of zero crossings while a low frequency signal should have a low number of zero crossings. ZCR denotes the rate of sign-changes of the signal during the duration of a particular frame. The mathematical definition of ZCR in a speech frame $x[n]$ is

$$ZCR[m] = \frac{1}{2M+1} \sum_{n=-\infty}^{\infty} \frac{1}{2} \left|\text{sgn}(x[n]) - \text{sgn}(x[n-1])\right| w[m-n]$$

where sgn() is the signum function (+1 for positive argument, -1 for negative argument). An example is shown in Figure 4.3.

4.3.4 Formants and Bandwidths

During speech, the human vocal tract changes all the time to allow us utter different phonemes and – as a consequence - syllables, words, and sentences. The change in the vocal tract shape results in increased spectral amplitude for certain frequencies. These frequencies are called formants. The number of formants in speech varies depending on the phonetic content of speech: vowels will almost always have four or more distinguishable formants; sometimes there are more than six (F1 to F7 or F8). F1 and F2 correlate well with the phonetic content of speech while higher formants describe mostly speaker characteristics. A formant is usually described by its center frequency and its bandwidth (and sometimes its amplitude).

Usually, some spectra envelope estimation method is applied on a speech frame to implicitly reveal formant information, with Linear Prediction (LP) being the most often used method. In this case, given that z_0 represents a complex root of the LP polynomial, the formant frequency F and bandwidth B are given by

$$F = \frac{F_s}{2\pi} \angle z_0$$

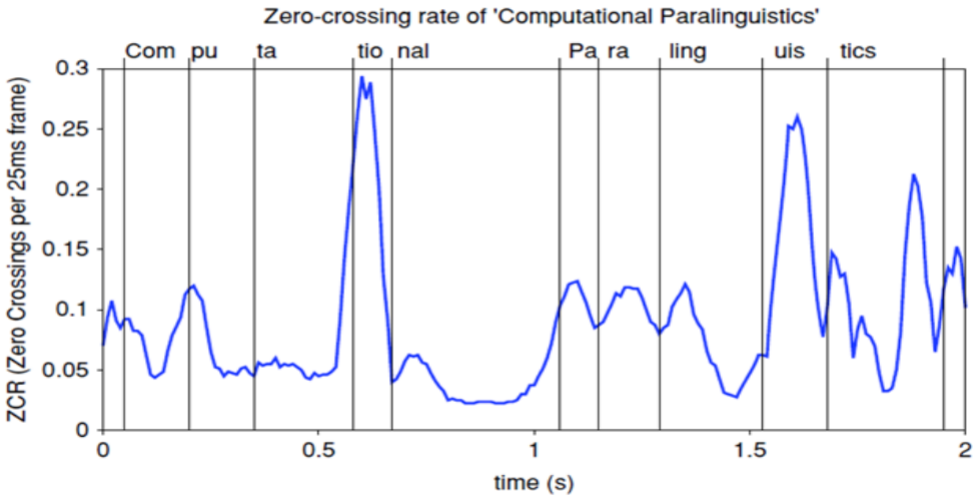


Figure 4.3: *Zero Crossings Rate contour of a speech waveform.*

$$B = -\frac{F_s}{\pi} \log |z_0|$$

An example of formant contours estimated from LP polynomial is shown in Figure 4.4.

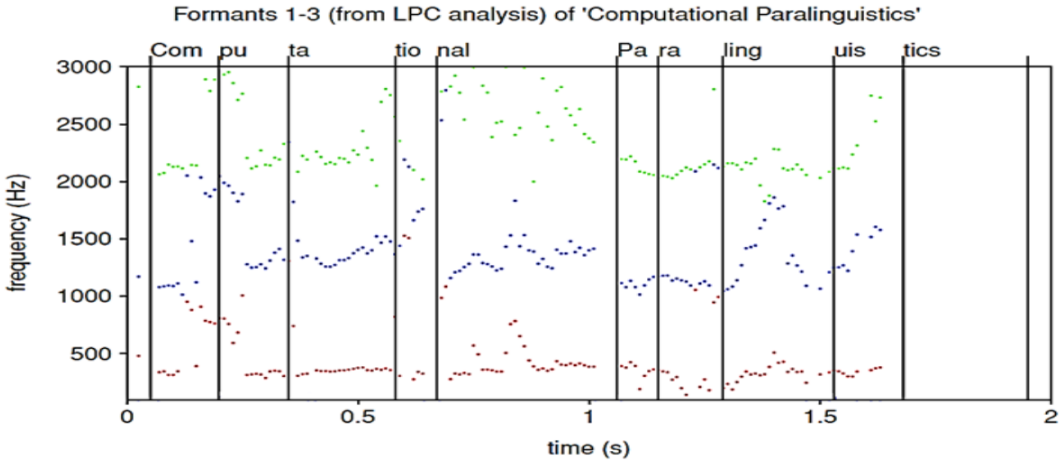


Figure 4.4: *Formant frequency contours.*

4.3.5 Intensity

The acoustic intensity of a sound is a physical quantity that can be defined as the average flow of energy (power) through a unit area measured in Watts per square meter. The human auditory system can detect a wide range of intensities, starting from 10^{-12} Watts per square meter and reaching up to 10 Watts per square meter. These two extremes correspond to the threshold of hearing and the threshold of pain, respectively. Expressed in dB, the intensity of a speech signal x[n] in air is defined as

$$I = 10 \log \frac{1}{P_0} [\frac{1}{N} \sum_{i=1}^{N} x^2[n]]$$

where P_0 equals 2×10^{-5} Pa.

4.3.6 Spectral Centroid

The spectral centroid (SC) is simply the center of gravity, or barycenter, of the spectrum of the speech frame. It is computed considering the spectrum as a distribution which values are the frequencies and the probabilities to observe these frequencies are the normalized amplitude values. Let X[k] be the N-point Fast Fourier Transform of the speech frame. Then the SC is defined as

$$C = \frac{\sum_{k=0}^{N-1} kX[k]}{\sum_{k=0}^{N-1} X[k]}$$

4.3.7 Spectral Spread

The spectral spread (SS) denotes the second central moment of the spectrum of the speech frame. The spectral spread describes the average deviation of the spectrum around its centroid, which is commonly associated with the bandwidth of the signal. Noise-like signals have usually a large spectral spread, while individual tonal sounds with isolated peaks will result in a low spectral spread. It is defined as

$$S = \sqrt{\frac{\sum_{k=0}^{N-1}(k-C)^2 X[k]}{\sum_{k=0}^{N-1} X[k]}}$$

4.3.8 Spectral Roll-off (90%)

The 90%-spectral roll-off is the frequency so that 90% of the signal energy is contained below this frequency.

4.3.9 Spectral Entropy

Spectral entropy (SE) of a signal is a measure of its spectral power distribution. The concept is based on the Shannon entropy, or information entropy, in information theory. The SE treats the signal's normalized power distribution in the frequency domain as a probability distribution, and calculates the Shannon entropy of it. The Shannon entropy in this context is the spectral entropy of the signal. We can define the spectral entropy in terms of power spectrum and probability distribution of a signal. If $S[k] = |X[k]|^2$ is the power spectrum of a speech frame, then P[k] is its probability distribution given by

$$P[k] = \frac{S[k]}{\sum_{i=0}^{N-1} S[i]}$$

Finally, the SE is given by

$$H = -\frac{\sum_{m=1}^{N} P[m] log_2 P[m]}{log_2 N}$$

4.3.10 Spectral Flux

The spectral flux (SF) is a measure of how quickly the power spectrum of a signal is changing, calculated by comparing the power spectrum for one frame against the power spectrum from the previous frame. More precisely, it is usually calculated as the 2-norm (also known as the Euclidean distance) between the two normalized spectra. SF can be calculated as

$$SF[k] = (\sum_{k=b_1}^{b_2} |X_t[k] - X_{t-1}[k]|^p)^{1/P}$$

where b_1 and b_2 are the band edges, in frequency bins, over which we calculate the SF. Usually, $b_1 = 0$ and b_2=N, and P=2.

4.3.11 Mel-frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCCs) model the spectral energy distribution in a perceptually meaningful way (take into account human perception on the construction of the frequency scale), that is, MFCCs is a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale. This mel-scale is defined as

$$Mel(f) = 2595 \log(1 + \frac{f}{700})$$

The mel-scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by assigning a perceptual pitch of 1000 mels to a 1000 Hz tone, 40 dB above the listener's threshold. Above about 500 Hz, increasingly large intervals are judged by listeners to produce equal pitch increments. The cepstrum is the Fourier Transform of the logarithm of the spectrum. The mel-cepstrum is the cepstrum computed on the mel-bands instead of the Fourier spectrum. The use of mel-scale allows to take better into account the low and mid-frequencies part of the signal. The MFCCs are the coefficients of the mel-cepstrum. These features are the most widely used audio features for paralinguistic event recognition such as emotion or personality recognition.

Technically, the process for extracting MFCCs is the following: we take the Fourier transform of a speech frame. Then, we map the power spectrum obtained above onto the mel-scale, using triangular overlapping windows or alternatively, cosine overlapping windows. Afterwards, we take the logs of the energies at each of the mel-frequencies and finally the discrete cosine transform is applied on the list of mel-log powers, as if it were a signal. The MFCCs are the amplitudes of the resulting signal. Example of MFCCs is shown in Fig. 4.5.

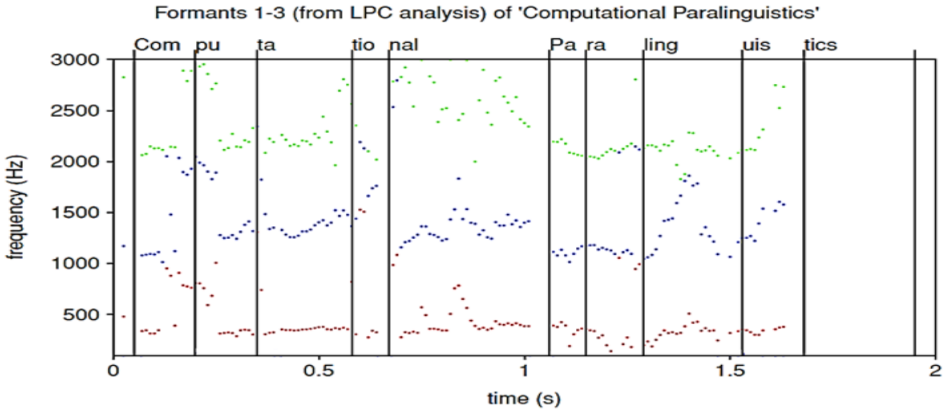


Figure 4.5: *MFCCs for a speech utterance.*

4.4 Conclusions

In this document, we have briefly described the features that can be used in a personality/emotion recognition system. A number of useful features have been described along with the statistical features that will be employed to derive the final features to be fed to the machine learning model. Both lists (features, statistics) are endless and should be constantly updated to improve performance.

Chapter 5

Personality Prediction Framework

This section provides an overview of our work, including dataset manipulation (as discussed in Section 3), code structure, machine learning techniques, and the integration of the PyQt5 library for the user interface and the presentation of results for each machine learning algorithm.

5.1 Dataset Manipulation

For each of the 640 speech clips we calculate 27 features. For each feature we obtain 7 statistics. Therefore our dataset consists of 640 rows and 189 (27 x 7) columns. Each speech clip is labeled as HIGH or LOW as follows: Every judge completes a personality questionnaire for all clips present in the corpus. Consequently, for a specific trait and a particular judge , a clip will fall either within the higher fifty percent of the scores assigned by the evaluator or in the lower fifty percent, thus allowing us to categorize a clip as "High" if it is in the upper half for the majority of the judges, or Low otherwise.

5.2 Classifiers

The **Classifier** class is designed for training and evaluating machine learning models on a given dataset. The code for this class is shown in Figure 5.1.

```
class Classifier:

    def __init__(self, data, labels, trait, splits = 5, scorers = {'roc_auc' : 'roc_auc'}, feature_selector = None):
        self.data = data
        self.labels = labels
        self.selector = feature_selector
        self.trait = trait
        self.splits = splits
        self.scorers = scorers
        self.model_dict = {
            ModelNames.SVM_MODEL : self.svmModel,
            ModelNames.LOG_REG : self.logRegression,
            ModelNames.GAUSS_NB : self.gaussianNB,
            ModelNames.DECISION_TREE : self.decisionTreeModel,
            ModelNames.KNN : self.knnModel,
            ModelNames.RNDFR : self.randomForestModel,
            ModelNames.ADABOOST : self.adaboostModel,
            ModelNames.XGBOOST : self.xgboostModel
        }
```

Figure 5.1: *Classifier class.*

data: input dataset for the training model.
labels: class for each training example in the dataset, for a specific trait.
trait: name of the trait we want to train our models.
splits: number of splits for cross-validation
scorers: scoring metrics to be used during model evaluation.
feature_selector: an optional feature selector object.
model_dict: dictionary that maps model names to their corresponding training methods.

The structure followed for training and evaluating machine learning models in our code exhibits a consistent and systematic approach. Each model adheres to the following structure:

1. **Data splitting**: The input data is initially divided into training and testing sets.
2. **Pipeline creation**: A machine learning pipeline is established consisting of a **feature scaler** and an optional **feature selector**.
3. **Hyperparameter optimization**: Hyperparameter optimization is performed.
4. **Cross-validation**: Cross-validation is performed to assess the model's performance on the training data.
5. **Model fitting and prediction**: the machine learning model is fitted to the training data using the pipeline. Predictions are then generated on the test data.
6. **Result reporting**: A set of performance metrics are computed to evaluate the model's effectiveness.

In Figure 5.2 we see an example for the case of the Logistic Regression model. The other models follow the same structure.

```
def logRegression(self):
    # split data
    x_train, x_test, y_train, y_test = train_test_split(self.data, self.labels, test_size=0.2, random_state=42)

    # create pipeline
    lr = LogisticRegression(class_weight=Parameters.LR_WEIGHT_CLASS, max_iter=Parameters.MAX_ITER)
    pipeline = Pipeline([('scaler', preprocessing.StandardScaler()),
                         ('select', self.selector),
                         ('lr', lr)])

    # optimize hyperparameters
    parameters = {'lr__C':np.logspace(-4,4,50)}
    clf = GridSearchCV(pipeline, parameters, n_jobs=4, cv=5, verbose=0, scoring='balanced_accuracy', refit=True)
    perform_cross_validation(pipeline, x_train, y_train, self.splits, self.scorers)

    # make predictions
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    y_pred_prob = clf.predict_proba(x_test)

    # report results
    accuracy = metrics.accuracy_score(y_test, y_pred)
    classification_metrics = metrics.classification_report(y_test, y_pred, digits=4)
    make_plots(y_test, y_pred, y_pred_prob[:,1], self.trait)
    print(classification_metrics)
    return accuracy, classification_metrics
```

Figure 5.2: *Logistic Regression Model.*

5.3 Feature Selectors

The **FeatureSelector** class is used for feature selection in machine learning pipelines, and is designed as shown in Figure 5.3.

```
class FeatureSelector:
    def __init__(self, selector_name = None, model_name = None, n_features = 5):
        self.n_features = n_features
        self.model_name = model_name
        self.selector_name = selector_name
        self.selector = None
        self.selector_dict = {
            FeatSelectorNames.RFE : self.recursiveFeatureSelector,
            FeatSelectorNames.SKB : self.selectKbestFeatureSelector,
            FeatSelectorNames.PCA : self.pcaFeatureSelector
        }
```

Figure 5.3: *FeatureSelector class.*

selector_name: name of the feature selector itself.
model_name: name of the machine learning model associated with the feature selector.
n_features: number of features to select.
selector_dict: dictionary that maps different feature selector names to their corresponding methods. It's used to select the appropriate feature selector method based on the provided **selector_name**.

5.4 User Interface

The user interface implemented in this work is illustrated in Figure 5.4.

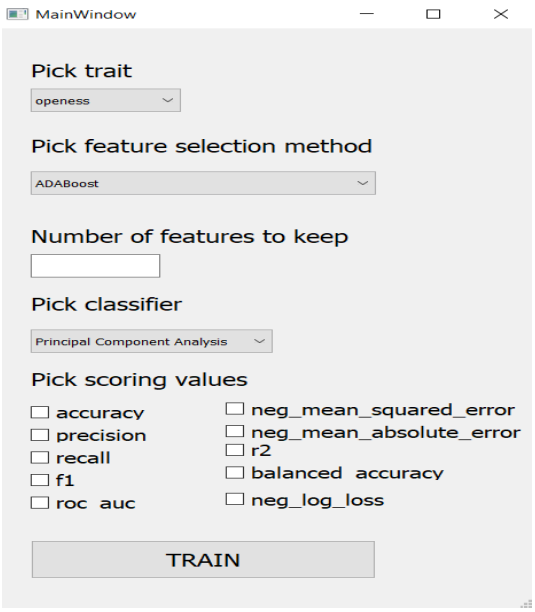


Figure 5.4: *Graphical User Interface.*

The GUI class is generated using **PyQt5**. This graphical user interface allows us to train any of the eight machine learning models specified in Section 3.6. We have the flexibility to select from five different traits for the dataset. Furthermore, we can specify the desired feature selection method and the number of features to retain. Finally, we can fine-tune the cross-validation process by choosing the scoring metrics that best suit our needs. This interface provides a user-friendly way to configure and initiate the training of various machine learning models with different settings and options.

Once we click the **train** button the code in Figure 5.5 is executed. The code has three main parts:

- **Data Acquisition and Preprocessing:** Obtains the dataset associated with the selected trait. Also, extracts the class labels from the dataset and removes **SF_min** column from the feature matrix because it has zero variance.
- **Scoring and Feature Selection:** Retrieves the selected scoring metrics for cross-validation. Also, it creates an instance of the FeatureSelector class with the selected feature selection method and number of features.
- **Model Training:** Initializes an instance of the Classifier class, passing in the dataset, labels, trait, scoring metrics, and feature selector. Then it calls the train_model method of the Classifier class to train the specified machine learning model.

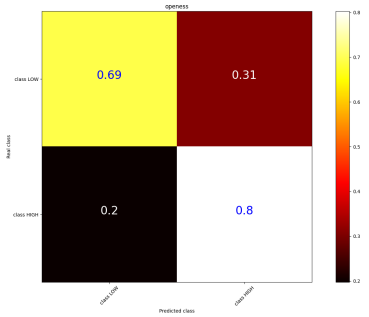
Overall, this method orchestrates the process of training a machine learning model based on user-defined parameters and options. It handles data retrieval, preprocessing, feature selection, and model training in response to user interactions with the GUI.

5.5 Experiments

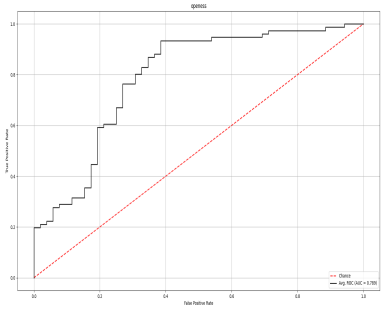
As we can see from the User Interface, there are many combinations of the options that are provided. In this section, we present the output of the UI when training a Support Vector Machine (SVM) model on the SSPNet dataset in Figures 5.6, 5.7, 5.8, 5.9, 5.10.

```python
def button_clicked(self):
    from dataAccess import DataAccessManager
    from utils import ModelNames, FeatSelectorNames
    from featureSelection import FeatureSelector
    from classifiers import Classifier

    trait = self.comboBox.currentText()
    model_name = self.comboBox_2.currentText()
    feat_selector_name = self.comboBox_3.currentText()
    n_feats = int(self.lineEdit.text())

    # obtain the data
    dtAccess = DataAccessManager()
    data = dtAccess.get_data(trait[0])
    labels = data['class']
    data = data.drop('class', axis=1)
    data = data.drop('SF_min', axis=1)

    # select scorer, classifier, feature selection method
    scorers = self.get_scoring_values()
    featSelector = FeatureSelector(feat_selector_name, model_name, n_feats)
    selector = featSelector.get_feature_selector()

    # train model
    classifier = Classifier(data, labels, trait, scorers=scorers, feature_selector=selector)
    classifier.train_model(model_name)
```

Figure 5.5: *Code executed when we train a model.*



(a) *Confusion matrix of an SVM model for predicting the Openness trait.*



(b) *ROC curve of SVM model for predicting the Openness trait.*

```
recall on train set using 5-Fold-CV: 0.779 (0.058)
roc_auc on train set using 5-Fold-CV: 0.832 (0.041)
Plotting curves and tables...

QCoreApplication::exec: The event loop is already running
Plotting ROC...

QCoreApplication::exec: The event loop is already running
              precision    recall  f1-score   support

           0     0.7059    0.6923    0.6990        52
           1     0.7922    0.8026    0.7974        76

    accuracy                         0.7578       128
   macro avg     0.7490    0.7475    0.7482       128
weighted avg     0.7571    0.7578    0.7574       128
```

(c) *5-fold cross-validation report: precision, recall, and f1-score.*

Figure 5.6: *Classification report of an SVM model for predicting the Openness trait.*

The output provides information about the SVM model's performance on the training set, including recall, and ROC-AUC, using 5-fold cross-validation. Additionally, it shows the ROC curve and a confusion matrix. Overall, this output is a comprehensive summary of the SVM model's training and evaluation results.

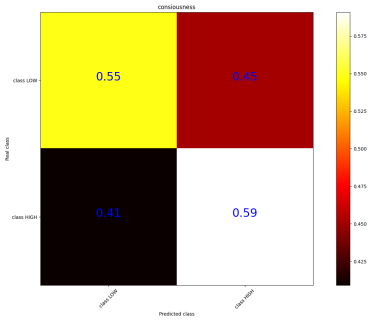
In the following tables, we provide a sample output for Recall (Sensitivity) and ROC-AUC for each of the available classifiers in Tables 5.1 to 5.8.

Table 5.1: Performance of the SVM classifier.

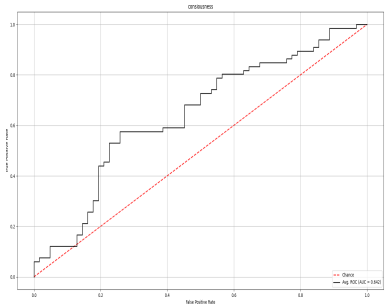
	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.614	0.606	0.711	0.654	0.566
AUC	0.788	0.640	0.746	0.696	0.609

Table 5.2: Performance of the Logistic Regression classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.705	0.55	0.638	0.627	0.571
AUC	0.788	0.589	0.694	0.691	0.610



(a) *Confusion matrix of an SVM model for predicting the Conscientiousness trait.*

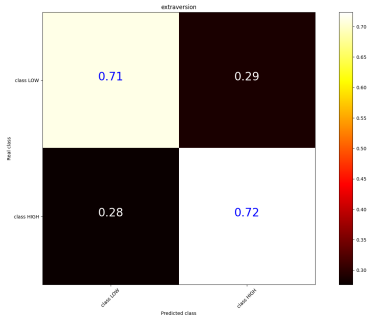


(b) *ROC curve of SVM model for predicting the Conscientiousness trait.*

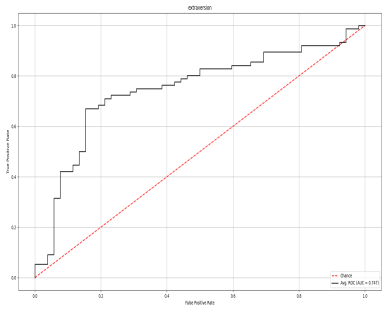
```
recall on train set using 5-Fold-CV: 0.722 (0.074)
roc_auc on train set using 5-Fold-CV: 0.737 (0.049)
Plotting curves and tables...

QCoreApplication::exec: The event loop is already running
Plotting ROC...

QCoreApplication::exec: The event loop is already running
              precision    recall  f1-score   support

           0     0.5574    0.5484    0.5528        62
           1     0.5821    0.5909    0.5865        66

    accuracy                         0.5703       128
   macro avg     0.5697    0.5696    0.5697       128
weighted avg     0.5701    0.5703    0.5702       128
```

(c) *5-fold cross-validation report: precision, recall, and f1-score.*

Figure 5.7: *Classification report of an SVM model for predicting the Conscientiousness trait.*



(a) *Confusion matrix of an SVM model for predicting the Extraversion trait.*

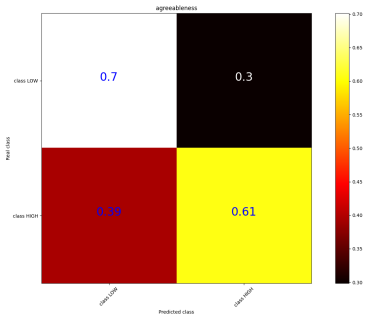


(b) *ROC curve of SVM model for predicting the Extraversion trait.*

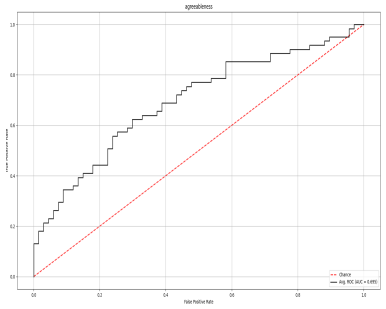
```
recall on train set using 5-Fold-CV: 0.710 (0.052)
roc_auc on train set using 5-Fold-CV: 0.762 (0.025)
Plotting curves and tables...

QCoreApplication::exec: The event loop is already running
Plotting ROC...

QCoreApplication::exec: The event loop is already running
              precision    recall  f1-score   support

           0     0.6379    0.7115    0.6727        52
           1     0.7857    0.7237    0.7534        76

    accuracy                         0.7188       128
   macro avg     0.7118    0.7176    0.7131       128
weighted avg     0.7257    0.7188    0.7206       128
```

(c) *5-fold cross-validation report: precision, recall, and f1-score.*

Figure 5.8: *Classification report of an SVM model for predicting the Extraversion trait.*



(a) *Confusion matrix of an SVM model for predicting the Agreeableness trait.*

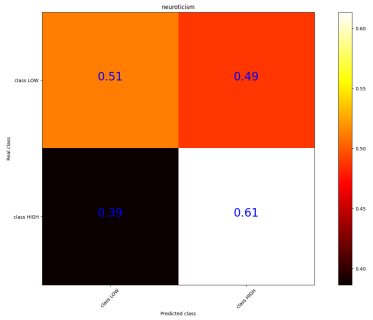


(b) *ROC curve of SVM model for predicting the Agreeableness trait.*

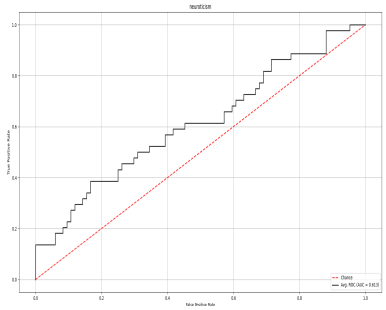
```
recall on train set using 5-Fold-CV: 0.622 (0.057)
roc_auc on train set using 5-Fold-CV: 0.735 (0.049)
Plotting curves and tables...

QCoreApplication::exec: The event loop is already running
Plotting ROC...

QCoreApplication::exec: The event loop is already running
              precision    recall  f1-score   support

           0     0.6620    0.7015    0.6812        67
           1     0.6491    0.6066    0.6271        61

    accuracy                         0.6562       128
   macro avg     0.6555    0.6540    0.6541       128
weighted avg     0.6558    0.6562    0.6554       128
```

(c) *5-fold cross-validation report: precision, recall, and f1-score.*

Figure 5.9: *Classification report of an SVM model for predicting the Agreeableness trait.*



(a) *Confusion matrix of an SVM model for predicting the Neuroticism trait.*



(b) *ROC curve of SVM model for predicting the Neuroticism trait.*

```
recall on train set using 5-Fold-CV: 0.619 (0.059)
roc_auc on train set using 5-Fold-CV: 0.573 (0.032)
Plotting curves and tables...

QCoreApplication::exec: The event loop is already running
Plotting ROC...

QCoreApplication::exec: The event loop is already running
              precision    recall  f1-score   support

           0     0.7167    0.5119    0.5972        84
           1     0.3971    0.6136    0.4821        44

    accuracy                         0.5469       128
   macro avg     0.5569    0.5628    0.5397       128
weighted avg     0.6068    0.5469    0.5577       128
```

(c) *5-fold cross-validation report: precision, recall, and f1-score.*

Figure 5.10: *Classification report of an SVM model for predicting the Neuroticism trait.*

Table 5.3: Performance of the GaussNB classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.734	0.606	0.703	0.556	0.512
AUC	0.802	0.611	0.729	0.618	0.516

Table 5.4: Performance of the Decision Tree classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.649	0.554	0.666	0.604	0.543
AUC	0.646	0.555	0.625	0.603	0.544

Table 5.5: Performance of the KNN classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.737	0.628	0.703	0.575	0.516
AUC	0.786	0.630	0.682	0.624	0.534

Table 5.6: Performance of the Random Forest classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.747	0.592	0.649	0.528	0.511
AUC	0.761	0.588	0.684	0.543	0.527

Table 5.7: Performance of the AdaBoost classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.632	0.555	0.596	0.592	0.535
AUC	0.674	0.544	0.608	0.637	0.580

Table 5.8: Performance of the XGBoost classifier.

	Openness	Conscientiousness	Extraversion	Agreeableness	Neuroticism
Recall	0.728	0.546	0.730	0.591	0.522
AUC	0.775	0.557	0.731	0.643	0.553

Chapter 6

Conclusion

In this thesis, we studied how speech signals can predict speaker personality using machine learning. We used the Scikit-Learn library to create machine learning algorithms that analyze speech clips and predict personality traits. We modeled speech signals using well-known features and used different feature selection methods to pick the most important information from these features. Additionally, we created a graphical user interface to simplify the process of training and testing.

Future studies can incorporate dimensionality reduction methods to make their models more efficient and easier to understand by simplifying the features they use. Moreover, advanced data cleaning and pre-processing techniques can be applied to ensure the highest data quality, further refining the robustness and accuracy of the models. Also, more features can be added to better model the speech signals, such as OPENSmile, that are related to paralinguistic tasks. Finally, one can allow the use of a microphone for real time personality assessment of a speaker who does not belong to the SSPNet dataset.

In summary, this thesis has provided a Python-based framework that offers a starting point for further study in personality perception through speech analysis.

References

[1] Clifford Nass, Youngme Moon, BJ Fogg, Byron Reeves, Chris Drye, "Can computer personalities be human personalities?".

[2] James S. UIeman, Leonard S. Newman, Gordon B. Moskowitz, "People as Flexible Interpreters: Evidence and Issues from Spontaneous Trait Inference."

[3] Allport GW, Cantril H (1934) Judging personality from voice.

[4] Addington DW (1968) The relationship of selected vocal characteristics to personality perceptions.

[5] Miron Zuckerman, Robert E. Driver (1989) What sounds beautiful is good: The vocal attractiveness stereotype.

[6] Brown B. Strong W. Rencher A.(1973) Fifty-four voices from two: the effects of simultaneous manipulations of rate, mean fundamental frequency, and variance of fundamental frequency on ratings of personality from speech.

[7] Gelareh Mohammadi and Alessandro Vinciarelli (2012) Automatic Personality Perception: Prediction of Trait Attribution Based on Prosodic Features.

[8] Robert R. McCrae and Oliver P. John (1992) An Introduction to the Five-Factor Model and Its Applications.

[9] Tupes, E. C. and Christal, R. E. (1961). Recurrent personality factors based on trait ratings.

[10] Amelang, Manfred Borkenau, Peter (1982) Über die faktorielle Struktur und externe Validität einiger Fragebogen-Skalen zur Erfassung von Dimensionen der Extraversion und emotionalen Labilität.

[11] McCrae, Robert R. Costa, Paul T. (1987) Validation of the five-factor model of personality across instruments and observers.

[12] Goldberg, L. R. (1989, June). Standard markers ofthe Big Five factor structure.

[13] Ostendorf F (1990) Sprache und Persönlichkeitsstruktur. Zur Validität des Fünf-Faktoren-Modells der Persönlichkeit

[14] Trapnell, P. D., & Wiggins, J. S. (1990). Extension ofthe Interpersonal Adjective Scales to include the Big Five dimensions of personality.

[15] McCrae. R. R., Costa, P. T., Jr. and Busch, C. M. (1986). Evaluating comprehensiveness in personality systems: The California Q-Set and the five-factor model.

[16] M Zuckerman, DM Kuhlman, C Camac (1988) What lies beyond E and N? Factor analyses of scales believed to measure basic dimensions of personality.

[17] RR McCrae (1987) Creativity, divergent thinking, and openness to experience.

[18] DW Fiske (1949) Consistency of the factorial structures of personality ratings from different sources.

[19] RR McCrae, PT Costa Jr (1989) Rotation to maximize the construct validity of factors in the NEO Personality Inventory.

[20] (2012) The interspeech 2012 speaker trait challenge.

[21] Michele Banko and Eric Brill (2001) Scaling to very very large corpora for natural language disambiguation.

[22] Amit Kumar Das Saikat Dutt, Subramanian Chandramouli, Machine Learning (2018).

[23] ROC Analysis figure

[24] R. McRae, The five-factor model of personality, P. Corr and G. Matthews, Eds., Cambridge, U.K.: Cambridge University Press, 2009, pp. 3-26.

[25] P. Ekman, W. Friesen, M. O'Sullivan and J. Scherer, "Relative importance of face, body, and speech in judgements of personality and affect," Personality Social Psychology, vol. 38, no. 2, pp. 270-277, 1980.

[26] B. Schuller, S. Steidl, A. Batliner, E. Noth, A. Vinciarelli, F. Burkhardt, R. V. Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi and B. Weiss, "The Interspeech 2012 speaker trait challenge," in Interspeech, 2012.

[27] G. Mohammadi, A. Vinciarelli and M. Mortillaro, "The Voice of Personality: Mapping Nonverbal Vocal Behavior into Trait Attributions," in 2nd Workshop on Social Signal Processing, Firenze, Italy, 2010.

[28] T. Polzehl, S. Moller and F. Metze, "Automatically assessing personality from speech," in IEEE International Conference on Semantic Computing, 2010.

[29] T. Polzehl, K. Schoenenberg, S. Moller, F. Metze, G. Mohammadi and A. Vinciarelli, "On Speaker-Independent Personality Perception and Prediction from Speech," in Interspeech, 2012.

[30] G. Mohammadi, A. Origlia, M. Filippone and A. Vinciarelli, "From speech to personality: mapping voice quality and intonation into personality differences," in ACM International Conference on Multimedia, 2012.

[31] G. Mohammadi and A. Vinciarelli, "Automatic Personality Perception: Prediction of Trait Attribution Based on Prosodic Features," IEEE Transactions on Affective Computing, vol. 3, no. 3, pp. 273-284, 2012.

[32] A. Ivanov and X. Chen, "Modulation Spectrum Analysis for Speaker Personality Trait Recognition," in Interspeech, Portland, 2012.

[33] A. V. Ivanov, G. Riccardi, A. J. Sporka and J. Franc, "Recognition of Personality Traits from Human Spoken Conversations," in Interspeech, Florence, Italy, 2011.

[34] C. Montacie and M. Caraty, "Pitch and Intonation Contribution to Speakers' Traits Classification," in Interspeech, Portland, USA, 2012.

[35] C. Chastagnol and L. Devillers, "Personality traits detection using a parallelized modified SFFS algorithm," in Interspeech, Portland, USA, 2012.

[36] M. H. Sanchez, A. Lawson, D. Vergyri and H. Bratt, "Multi-System Fusion of Extended Context Prosodic and Cepstral Features for Paralinguistic Speaker Trait Classification," in Interspeech, Portland, USA, 2012.

[37] M.-H. Su, C.-H. Wu, K.-Y. Huang, Q.-B. Hong and W. H-M., "Personality Trait Perception from Speech Signals using Multiresolution Analysis and Convolutional Neural Networks," in APSIPA, Malaysia, 2016.

[38] G. An and R. Levitan, "Lexical and Acoustic Deep Learning Model for Personality Recognition," in Interspeech, Hyderabad, India, 2018.

[39] L. Gilpin, D. Olson and T. Alrashed, "Perception of Speaker Personality Traits Using Speech Signals," in CHI Conference on Human Factors in Computing Systems, Monteal, Canada, 2018.

[40] J. Yu, K. Markov and A. Karpov, "Speaking Style Based Apparent Personality Prediction," in International Conference on Speech and Computer, Instanbul, Turkey, 2019.