# AUTISM SPECTRUM DISORDER DETECTION USING MACHINE LEARNING

Thesis submitted in fulfilment of the requirement for the degree

Of

Bachelor of Technology (B.Tech.)

In

Electronics and Communication Engineering

By

| | |
|---|---|
| **Abhijeet Sarkar** | **500221010002** |
| **Ashmita Saha** | **500221020017** |
| **Bhaskar Bhattacharjee** | **500221010020** |
| **Bishnupada Sahoo** | **500221010021** |
| **Pradipta Banerjee** | **500221010042** |
| **Rounak Bakshi** | **500221010052** |

*Under the guidance of*

**Dr. Suparna Biswas**

(Associate Professor, ECE Department)

Guru Nanak Institute of Technology, Kolkata- 700114

# CERTIFICATE OF RECOMMENDATION

This is to certify that the work in preparing the dissertation entitled "**AUTISM SPECTRUM DISORDER DETECTION USING MACHINE LEARNING**", has been carried out by Abhijeet Sarkar, Ashmita Saha, Bhaskar Bhattacharjee, Bishnupada Sahoo, Pradipta Banerjee and Rounak Bakshi under our supervision and may be accepted as a partial fulfilment for the requirement of the degree leading to Bachelor of Technology in Electronics and Communication Engineering Department at Guru Nanak Institute of Technology.

**Dr. Suparna Biswas**
**Associate Professor, ECE Department**
**GNIT, Kolkata**

**Dr. Avali Banerjee**
**HOD, ECE Department**
**GNIT, Kolkata**

# DECLARATION

We, hereby declare that the project work entitled “**AUTISM SPECTRUM DISORDER DETECTION USING MACHINE LEARNING**” is a record of original work done by us under the guidance of Dr. Suparna Biswas, Department of Electronics and Communication Engineering, Guru Nanak Institute of Technology, and this project is submitted in the partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Electronics and Communication Engineering. The results embodied in this thesis have not been submitted to any other university or institute for the award of any degree.

## <u>Team Members</u>

**Abhijeet Sarkar (500221010002)**

**Ashmita Saha (500221020017)**

**Bhaskar Bhattacharjee (500221010020)**

**Bishnupada Sahoo (500221010021)**

**Pradipta Banerjee (500221010042)**

**Rounak Bakshi (500221010052)**

## ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our project guide Dr. Suparna Biswas for their able guidance and support in completing our project.

We would also like to extend our gratitude to our HOD Ma’am Dr. Avali Banerjee for providing us with all the facility that was required.

## TABLE OF CONTENTS

# 1. Abstract

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition characterized by deficits in social interaction, communication, and behavior. Early and accurate detection is essential for effective intervention, yet traditional diagnostic methods can be time-consuming and require clinical expertise. This study investigates a multimodal machine learning approach for ASD detection using two complementary data sources: structured diagnostic questionnaires and neuroimaging data from the ABIDE (Autism Brain Imaging Data Exchange) repository. A comprehensive pre-processing pipeline was employed, including data cleaning, encoding, normalization, skull-stripping, smoothing, and dimensionality reduction via Principal Component Analysis (PCA), to ensure data quality and model readiness. Classical machine learning algorithms such as Logistic Regression, Support Vector Machines, and k-Nearest Neighbours were trained and evaluated on both datasets independently.

Logistic Regression emerged as the top performer, achieving a remarkable 99.2% accuracy on the questionnaire dataset and 80.7% accuracy on the ABIDE dataset, demonstrating strong generalizability across behavioural and neurobiological modalities. The integration of these two data types provided deeper insights into the manifestation of ASD, capturing both observable traits and brain-based markers. The results highlight the effectiveness of proper pre-processing, model selection, and dimensionality reduction in improving performance, especially in data-constrained scenarios.

Despite challenges such as limited sample size, data heterogeneity, and computational overhead, this study underscores the feasibility of scalable, interpretable, and clinically relevant machine learning tools for ASD detection. Future work will explore the use of deep learning models on neuroimaging data, the fusion of multimodal features for unified diagnosis, and the development of real-time diagnostic systems for broader clinical deployment.

**Keywords:**

Autism Spectrum Disorder (ASD), Machine Learning, Neuroimaging, ABIDE Dataset, Diagnostic Questionnaires, Logistic Regression, Support Vector Machines, Random Forest, Decision Trees, KNN, Dimensionality Reduction, Principal Component Analysis (PCA), Multimodal Analysis, fMRI, Feature Selection, Clinical Decision Support, Medical AI, Early Diagnosis, Data Pre-processing, Model Interpretability, Computational Psychiatry

# 2. Introduction

## 2.1 Background on Autism Spectrum Disorder (ASD)

Autism Spectrum Disorder (ASD) is a neurological condition that affects how a person thinks, communicates, interacts socially, and processes sensory information. The term "spectrum" reflects the wide range of characteristics and abilities individuals with autism may have. While some may face significant challenges in their day-to-day existence, others may have exceptional skills and live independently.

ASD usually starts to show up in early childhood, and by the time a child is two or three years old, there are often clear signs. Some common signs are trouble communicating verbally and nonverbally, doing the same things over and over, having few interests, and having trouble getting along with others. It's important to understand and help people with ASD because they are all different.

Early diagnosis is very important. If kids with ASD get the right kind of help and support, like occupational therapy, behavioural therapy, speech therapy, or educational support, they are more likely to learn important skills and become independent over time. Support is still very important in adolescence and adulthood; it helps people with their mental health, careers, and social lives.

Although the exact cause of ASD is still unknown, researchers think that a combination of environmental and genetic factors contributes to the disorder. Parenting styles or individual decisions are not to blame. As ASD has become more widely recognised and diagnosed in the past few decades, there are now more resources and support networks available.

Early assistance in the development of vital life skills is very beneficial for children with ASD. Personalised treatment plans and educational support can make a significant difference. Importantly, the increasing acceptance of the abilities and contributions of people on the spectrum is promoting an open culture that values neurodiversity. Acceptance and inclusion are becoming increasingly important in today's society. Numerous communities and advocacy organisations are trying to change the conversation away from "curing" autism and towards embracing neurodiversity. Individuals with ASD can contribute unique perspectives, innovative concepts, and useful abilities, and they can lead fulfilling lives in the correct setting.

In conclusion, autism spectrum disorder is a collection of differences that impact how individuals view and engage with the world instead of a single illness. People with ASD can succeed in every aspect of life if they are understood, accepted, and given the proper support. Making the world more inclusive and supportive begins with understanding ASD. We can all help people on the autism spectrum live better lives if we have patience, empathy, and knowledge.

## 2.2 Importance of Early Detection

Early detection of Autism Spectrum Disorder (ASD) is one of the most critical factors in ensuring a better quality of life for individuals on the spectrum. Identifying signs of autism at an early age, typically between 18 months and 3 years, allows for timely intervention, which is essential during the critical period of brain development. Children can get the help they require during the most critical phases of brain development if the early symptoms of autism, such as delayed speech, limited eye contact, or repetitive behaviours, are identified.

During early childhood, the brain is especially adaptable. A child's brain is most capable to learn new behaviours and skills during this time, which is frequently referred to as the "critical window." If ASD is identified early, targeted therapies can begin when they are most effective. According to research, learning, social interactions, and communication skills can all significantly improve with early intervention.

Additionally, early detection gives carers and parents more power. It allows them to learn how to support their child's special requirements, access the right services, and understand their child's condition. Children with ASD may not receive the necessary support if they are not diagnosed early, and parents may feel overwhelmed or confused if they are unaware of the causes of their child's behaviour.

Schools, medical professionals, and therapists all gain from early identification. Early diagnosis allows professionals to create specialised education and support plans for the child. The child benefits from these specialised methods in both educational and social environments. It's also critical to remember that early detection decreases long-term difficulties. Early support increases a child's likelihood of becoming self-sufficient, confident, and emotionally resilient. As they age, they might also need less intensive support.

In conclusion, early identification of ASD is transformative rather than just helpful. It creates the basis for a happy life, boosts the likelihood of positive development, and provides access to early support.

Early identification of ASD can alter a child's entire developmental trajectory in addition to providing a head start. It makes it possible for professionals and families to interact right away, setting the foundation for development, education, and deep connections with society.

## 2.3 Motivation for using Machine Learning

Machine learning (ML) has emerged as an increasingly essential asset in the healthcare sector, enabling the analysis of extensive and intricate datasets, uncovering concealed patterns, and making highly accurate predictions. In relation to Autism Spectrum Disorder (ASD), ML offers a distinctive opportunity to revolutionize the conventional diagnostic approach, which is frequently subjective, labour-intensive, and time-consuming.

One significant reason for implementing ML in ASD detection is its capacity to facilitate early screening through automation and standardization. Traditional techniques heavily depend on expert interpretation of behavioural symptoms, which can differ significantly among clinicians. In contrast, machine learning can analyze objective data - such as responses to questionnaires, developmental history, and even video or audio signals - to produce predictions based on learned patterns from previously classified cases.

ML models, including decision trees, support vector machines (SVM), random forests, and deep learning networks, can be trained on labelled ASD datasets to differentiate between individuals with ASD and those without. These models can also pinpoint which characteristics (e.g., lack of eye contact, delayed speech) are most indicative of an ASD diagnosis, thereby enhancing the refinement of current screening tools.

The scalability of ML represents another significant benefit. Once trained and validated, these models can be integrated into mobile applications, online platforms, or diagnostic software, thereby making screening more accessible in under-resourced or rural regions. Additionally, ML models function efficiently, providing predictions within seconds and alleviating the workload of healthcare professionals.

Moreover, ML paves the way for personalized diagnostics by continuously learning from new data and adjusting to specific population traits. As more data becomes available, models can be updated to reflect evolving diagnostic trends and enhance generalizability across diverse populations.

In conclusion, the rationale for employing machine learning in ASD detection is rooted in its ability to support early and accurate diagnosis.

## 2.4 Project Objectives

The main aim of this initiative is to utilize machine learning algorithms for the early identification of Autism Spectrum Disorder through the analysis of real-world datasets. This encompasses the design, implementation, and assessment of predictive models capable of categorizing individuals as either likely or unlikely to have ASD based on behavioural, demographic, or developmental characteristics.

The project is directed by the following specific goals:

- To gather and pre-process dependable ASD datasets that encompasses pertinent features such as age, gender, responses to screening questionnaires, and developmental history.

- To conduct exploratory data analysis (EDA) to comprehend the structure, distribution, and correlations within the data, which will guide feature selection and model development.

- To train and evaluate various machine learning algorithms, including logistic regression, decision trees, random forests, and support vector machines, and to compare their effectiveness using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

- To pinpoint the most critical predictive features that influence ASD classification, which may assist in enhancing current screening tools or inform the creation of new ones.

- To assess the practical implications and constraints of employing machine learning in a real-world clinical environment, taking into account ethical considerations, data privacy, and interpretability.

By fulfilling these goals, the project aims to advance the integration of artificial intelligence into healthcare, particularly in domains that can benefit from early intervention and improved accessibility, such as autism diagnosis.

# 3. Literature Review on detection of Autism Spectrum Disorder using Machine Learning

## Overview

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition characterized by challenges in social interaction, repetitive behaviors, and restricted interests. Traditional diagnostic methods are often subjective and time-intensive. As a result, machine learning (ML) has emerged as a promising approach to support early, scalable, and accurate ASD detection. ML techniques have shown high potential when applied to behavioral assessments, EEG recordings, and neuroimaging data, providing automated, objective, and efficient diagnostic support.

## 3.1 Machine Learning in ASD Detection

Machine learning enables the development of predictive models that can recognize ASD-related patterns from structured datasets, such as EEG signals, behavioral questionnaire responses, or brain imaging data. These models learn from labeled training data and can differentiate ASD from non-ASD individuals with high accuracy, making them suitable for early screening and diagnosis.

## 3.2 MRI-Based Applications

Magnetic Resonance Imaging (MRI) provides detailed anatomical and functional information about the brain, making it highly valuable in ASD research. Machine learning models applied to MRI data can detect structural abnormalities, altered brain volumes, and atypical connectivity patterns often associated with autism.

**Notable Studies and Researchers:**

- **Ghiassian et al. (2016)[5]:** Utilized structural and functional MRI data from the ABIDE dataset combined with demographic data. Their SVM-based model achieved around 65% accuracy in classifying ASD and neurotypical individuals.
- **Arbabshirani et al. (2013)[6]:** Applied Random Forest to structural MRI data, identifying differences in gray and white matter volumes with classification accuracy of approximately 85%.
- **Chen et al. (2015)[7]:** Used Random Forest on ABIDE functional connectivity data, achieving 90.8% accuracy with high sensitivity (89%) and specificity (93%).
- **Wolfers et al. (2015)[8]:** Employed kNN with features like cortical thickness and gray matter volume, showing moderate classification success and emphasizing the importance of careful feature selection.

- **Heinsfeld et al. (2018)[9]:** Leveraged PCA for dimensionality reduction on MRI data before applying machine learning models, improving accuracy by retaining key features and reducing noise.

## 3.3 Common Machine Learning Algorithms

**Support Vector Machines (SVM)**

- Effective in high-dimensional binary classification.
- Successfully applied by **Ghiassian et al. (2016)[5]** on MRI data with promising results.

**Random Forest (RF) and Decision Trees (DT)**

- Robust ensemble methods capable of handling large and diverse datasets.
- Demonstrated high accuracy in studies by **Arbabshirani et al. (2013)[6]** and **Chen et al. (2015)[7]**.

**k-Nearest Neighbors (kNN)**

- Classifies based on similarity with neighboring data points.
- Applied by **Wolfers et al. (2015)[7]** with MRI-derived features.

**Artificial Neural Networks (ANN)**

- While more often applied to EEG data, ANN models have been explored with neuroimaging as well.

## 3.4 Feature Engineering and Preprocessing

**Dimensionality Reduction**

- **Principal Component Analysis (PCA):** Used by **Heinsfeld et al. (2018)[9]** to reduce complexity in MRI data while preserving key features.

**Noise Reduction Techniques**

- **Independent Component Analysis (ICA):** Removes artifacts such as eye blinks in EEG.
- **Wavelet Decomposition:** Separates EEG signals into frequency bands, allowing removal of noise and reconstruction of clean signals.

## Conclusion

Machine learning has emerged as a powerful tool in the early detection and diagnosis of Autism Spectrum Disorder. Studies by Grossi[1], Jamal[2], Ghiassian[5], Bosl[3], Arbabshirani[6], and others have demonstrated the efficacy of ML algorithms such as SVM, RF, kNN, and ANN in analyzing EEG and MRI data. These techniques provide objective, interpretable, and scalable alternatives to traditional diagnostic approaches. Continued research and refinement in this area hold promise for enhancing early ASD detection and enabling timely interventions.

# 4. Datasets used

## 4.1 Dataset Overview

Two distinct but complementary datasets were used in our project on Autism Detection using Machine Learning: the *ABIDE phenotypic dataset* and a publicly available *autism screening dataset*. These datasets, though differing in structure and content, together provided a robust foundation for exploring autism spectrum disorder (ASD) classification from both clinical and behavioural perspectives. While the ABIDE dataset offered rich phenotypic data linked to MRI imaging, the screening dataset supplied accessible, structured questionnaire data commonly used in early autism detection. The combination allowed for a dual approach—leveraging both medically-validated information and real-world screening tools—to train, validate, and compare machine learning models aimed at distinguishing individuals with ASD from typically developing individuals.

## 4.2 Autism Screening Questionnaire Dataset

The second dataset (autism_data.csv) is a tabular dataset that resembles standardized autism screening forms. It includes individual-level responses to a series of binary or multiple-choice questions, typically administered in self-report or caregiver-report formats. These questions are designed to identify behavioural patterns commonly associated with autism, such as social withdrawal, difficulty with communication, and repetitive behaviours.

Alongside these questionnaire responses, the dataset also includes demographic fields such as age, gender, country of residence, and education level, as well as a binary target label indicating whether a subject was classified as autistic or not. Compared to the ABIDE dataset, this data is more simplified but allows for rapid prototyping and evaluation of machine learning models using structured, easy-to-obtain data. It is particularly relevant for real-world applications where clinical or imaging data may not be readily available.

In the project, this dataset was instrumental in testing several baseline classifiers and tuning algorithms based on common metrics like accuracy, precision, recall and f1-score. Its clean structure also made it suitable for exploratory data analysis and feature selection.

The dataset contains information related to autism screening, with the following 21 columns:

1) **A1_Score to A10_Score**: These are binary (0 or 1) responses to ten screening questions from the Autism Spectrum Quotient (AQ-10) test. A response of 1 typically indicates behavior associated with autism.

2) **age**: The age of the individual (in years).
3) **gender**: The gender of the individual, typically 'm' for male and 'f' for female.
4) **ethnicity**: The individual's ethnic background (e.g., 'White-European', 'Latino', 'Asian', etc.).
5) **jundice**: Whether the individual had neonatal jaundice ('yes' or 'no').
6) **austim**: Whether the individual has a family history of autism ('yes' or 'no').
7) **contry_of_res**: Country of residence of the individual (note the typo: "contry" instead of "country").
8) **used_app_before**: Whether the individual has used a screening app before ('yes' or 'no').
9) **result**: The screening result score from the AQ-10 test (a numerical value).
10) **age_desc**: Description of the age group (e.g., '18 and more').
11) **relation**: The relationship of the person filling out the form to the individual being screened (e.g., 'Self', 'Parent', etc.).
12) **Class/ASD**: The final classification based on the test — 'YES' if the person is likely to have autism, 'NO' otherwise.

## 4.3 ABIDE Phenotypic Dataset

The ABIDE phenotypic dataset (Phenotypic_V1_0b_preprocessed1.xlsx) is part of the larger Autism Brain Imaging Data Exchange project. It includes subject-level metadata corresponding to participants whose neuroimaging data was collected from 17 international sites. Although this project focused on the phenotypic (non-imaging) data, the dataset remains closely tied to fMRI-based research in autism.

Each row in the dataset represents an individual participant and contains a wide array of variables. These include demographic information such as age at scan, sex, and handedness; diagnostic classification (Autism or Control); cognitive assessments like Full-Scale IQ (FIQ), Verbal IQ (VIQ), and Performance IQ (PIQ); and various clinical and behavioural scores derived from standardized instruments such as the ADOS (Autism Diagnostic Observation Schedule), SRS (Social Responsiveness Scale), and SCQ (Social Communication Questionnaire). In addition, quality control metrics for scan reliability—such as frame wise displacement and root mean square (RMS) motion estimates—are included for filtering participants during data pre-processing.

This dataset enabled a detailed understanding of the characteristics typically associated with autism and was used in the project to construct feature sets for machine learning classifiers. It was particularly useful for examining how demographic and cognitive variables could influence ASD diagnosis and how predictive such features could be when used in isolation or combination.

The dataset contains 106 fields, many of these are clinical, diagnostic, or quality control measures.

1) Subject Identifiers

- **i, Unnamed: 0, X**: Internal or indexing fields (can usually be ignored).
- **SUB_ID**: Subject ID.
- subject: Often duplicate or formatted version of SUB_ID.
- **SITE_ID**: Data acquisition site (e.g., NYU, UCLA).
- **FILE_ID**: File name identifier.

2) Diagnosis and Demographics

- **DX_GROUP**: Diagnosis group (1 = Control, 2 = Autism).
- **DSM_IV_TR**: DSM-IV-TR classification.
- **AGE_AT_SCAN**: Age in years at time of scanning.
- **SEX**: 1 = Male, 2 = Female.
- **HANDEDNESS_CATEGORY**: Right, Left, Ambidextrous.
- **HANDEDNESS_SCORES**: Numeric handedness score.

3) IQ Scores and Test Types

- **FIQ, VIQ, PIQ**: Full, Verbal, and Performance IQ.
- **FIQ_TEST_TYPE, VIQ_TEST_TYPE, PIQ_TEST_TYPE**: Corresponding test types used.

4) ADI-R (Autism Diagnostic Interview – Revised) Scores

- **ADI_R_SOCIAL_TOTAL_A**
- **ADI_R_VERBAL_TOTAL_BV**
- **ADI_RRB_TOTAL_C**: Restricted/repetitive behavior.
- **ADI_R_ONSET_TOTAL_D**: Age of symptom onset.
- **ADI_R_RSRCH_RELIABLE**: Whether interview was research-reliable.

5) ADOS (Autism Diagnostic Observation Schedule) Scores

- **ADOS_MODULE**: Module used (depends on age/verbal ability).
- **ADOS_TOTAL, ADOS_COMM, ADOS_SOCIAL, ADOS_STEREO_BEHAV**: Scores across domains.
- **ADOS_RSRCH_RELIABLE**: Research-reliable status.

- **ADOS_GOTHAM_SOCAFFECT, ADOS_GOTHAM_RRB, ADOS_GOTHAM_TOTAL, ADOS_GOTHAM_SEVERITY**: Gotham algorithm scores.

6) SRS (Social Responsiveness Scale) Scores
   - **SRS_VERSION**: Version used.
   - **SRS_RAW_TOTAL**: Total raw score.
   - **SRS_AWARENESS, SRS_COGNITION, SRS_COMMUNICATION, SRS_MOTIVATION, SRS_MANNERISMS**: Subscale scores.

7) Other Behavioral Assessments
   - **SCQ_TOTAL**: Social Communication Questionnaire total.
   - **AQ_TOTAL**: Autism Quotient score.

8) Medical and Comorbidity
   - **COMORBIDITY**: Comorbid conditions.
   - **CURRENT_MED_STATUS**: Medication usage status.
   - **MEDICATION_NAME**: Name of medications.
   - **OFF_STIMULANTS_AT_SCAN**: Was off stimulants during scan?

9) Vineland Adaptive Behavior Scales (Used to assess personal and social skills)
   - **VINELAND_RECEPTIVE_V_SCALED, ..., VINELAND_COPING_V_SCALED**: Scaled subdomain scores.
   - **VINELAND_COMMUNICATION_STANDARD, VINELAND_DAILYLVNG_STANDARD, VINELAND_SOCIAL_STANDARD, VINELAND_ABC_STANDARD**: Standard scores.
   - **VINELAND_SUM_SCORES**: Sum of scaled scores.
   - **VINELAND_INFORMANT**: Person reporting the scores (e.g., parent).

10) WISC-IV (Wechsler Intelligence Scale for Children - Fourth Edition)
    - **WISC_IV_VCI, WISC_IV_PRI, WISC_IV_WMI, WISC_IV_PSI**: Composite scores.
    - Subtest scaled scores like:
      - **WISC_IV_SIM_SCALED, VOCAB_SCALED, INFO_SCALED, ...**
      - **WISC_IV_BLK_DSN_SCALED, MATRIX_SCALED, DIGIT_SPAN_SCALED, ...**

11) Scan Status and Motion
    - **EYE_STATUS_AT_SCAN**: Eyes open/closed during scan.
    - **AGE_AT_MPRAGE**: Age at T1-weighted scan.
    - **BMI**: Body Mass Index.

12) Quality Metrics (Anatomical and Functional MRI)

- **anat_cnr, anat_efc, anat_fber, anat_fwhm, anat_qi1, anat_snr**: Structural scan quality.
- **func_efc, func_fber, func_fwhm, func_dvars, func_outlier, func_quality**: Functional scan quality.
- **func_mean_fd, func_num_fd, func_perc_fd**: Framewise displacement (motion).
- **func_gsr**: Whether global signal regression was applied.

13) Quality Control Ratings

- **qc_rater_1, qc_notes_rater_1, ... qc_anat_rater_2/3, qc_func_rater_2/3**: Manual ratings.
- **qc_anat_notes_*, qc_func_notes_***: QC notes from different raters.

14) Final Inclusion Status

- **SUB_IN_SMP**: Whether the subject was included in the sample used in publication.

## Combined Use

By combining the clinical depth of the ABIDE phenotypic data with the practical accessibility of the autism screening data, this project was able to evaluate autism detection from two complementary angles: one rooted in comprehensive, medically-reviewed data, and the other based on everyday behavioural screening tools. This approach not only strengthened the reliability of the classification models but also showcased the potential for scalable autism detection solutions that bridge clinical research and real-world usability.

# 5. Methodology

## 5.1 Overview of Project Workflow

Data collection, pre-processing, feature selection, model training, and evaluation are all part of the project's organized pipeline. Two different datasets—a neuroimaging dataset (ABIDE) and a text-based dataset with questionnaire responses—were employed. Every dataset needed a different approach to modelling and pre-processing. The high-level process consists of:

- **Data Collection**: Gathering data from two primary sources — a structured questionnaire dataset and the ABIDE (Autism Brain Imaging Data Exchange) repository.
- **Pre-processing**: Customized pre-processing for each dataset type, including cleaning, normalization, and skull-stripping for MRI data.
- **Feature Engineering and Dimensionality Reduction**: Extracting meaningful features and reducing dimensionality to avoid overfitting and computational overhead.
- **Model Building**: Training classical machine learning models like Logistic Regression, SVM, and KNN tailored to each dataset.
- **Evaluation and Analysis**: Using performance metrics and visual tools like confusion matrices to assess model effectiveness.

This modular approach ensures flexibility, scalability, and robust handling of heterogeneous data sources.

## 5.2 Data Pre-processing Techniques

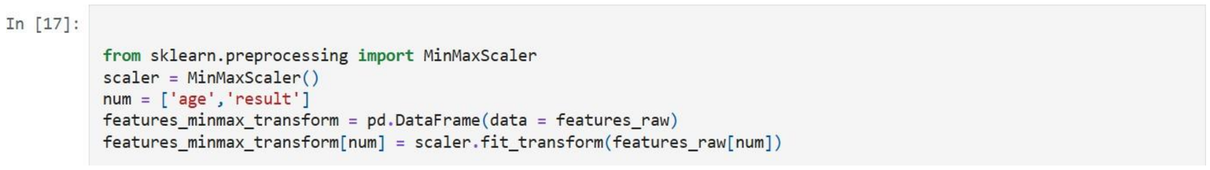
### 5.2.1 Screening Questionnaire Dataset:

The study's text dataset comes from standardized diagnostic questionnaires that evaluate demographic and behavioural indicators linked to autism spectrum disorder (ASD). Age, gender, ethnicity, country of residence, history of familial ASD, and screening results are among the characteristics included in this dataset. Such data must go through a methodical pre-processing pipeline to be cleaned, consistent, and machine-readable before it can be used for machine learning.

1) **Handling Missing Values:**
    - **Technique:** The code uses data.dropna(inplace=True) to remove rows with missing values.
    - **Reasoning:** Missing values can cause issues with many machine learning algorithms. Removing them is a simple way to handle this problem.

2) **Feature Scaling:**

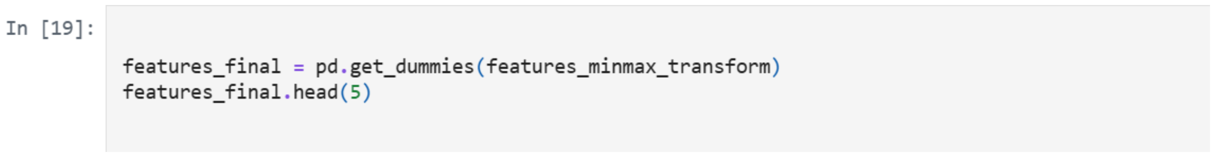
- **Technique:** MinMaxScaler is used to scale numerical features ('age' and 'result') to a range between 0 and 1.
- **Reasoning:** This ensures that features with different scales (e.g., age and result) have a similar impact on the model. It can improve the performance of algorithms like SVM and KNN.



```
In [17]:
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
num = ['age','result']
features_minmax_transform = pd.DataFrame(data = features_raw)
features_minmax_transform[num] = scaler.fit_transform(features_raw[num])
```

3) **One-Hot Encoding:**

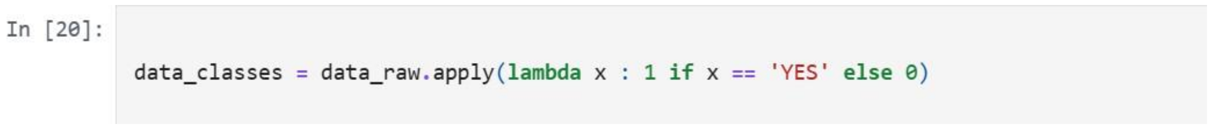
- **Technique:** pd.get_dummies() is used to convert categorical features into numerical representations using one-hot encoding.
- **Reasoning:** Machine learning algorithms typically work with numerical data. One-hot encoding creates new binary columns for each category, preventing the model from misinterpreting categorical values as ordinal.



```
In [19]:
features_final = pd.get_dummies(features_minmax_transform)
features_final.head(5)
```

4) **Label Encoding:**

- **Technique:** The target variable ('Class/ASD') is converted from 'YES' and 'NO' to 1 and 0, respectively, using a lambda function.
- **Reasoning:** Similar to one-hot encoding, this converts the target variable into a numerical format suitable for machine learning models.



```
In [20]:
data_classes = data_raw.apply(lambda x : 1 if x == 'YES' else 0)
```

The pre-processing of the text dataset is a crucial step that changes raw, unstructured data into a well-formed and numerical form suitable for machine learning models. This rigorous preparation improves data quality, reduces bias, and ensures better generalization of the trained models.

### 5.2.2 ABIDE Dataset: Pre-processing and Feature Preparation

The Autism Brain Imaging Data Exchange (ABIDE) dataset used in this study was obtained from the Pre-processed Connectomes Project (PCP), specifically the 'rois_aal' derivative. This version of the dataset provides resting-state fMRI data that has already undergone a standardized pre-processing pipeline, ensuring consistency and reproducibility across studies.

#### Pre-processing Performed by PCP

The following pre-processing steps were completed by the PCP team:

- **Skull-stripping**: This involves removing non-brain tissues such as the skull, scalp, and other surrounding structures from anatomical scans, focusing the analysis solely on the brain.
- **Spatial Normalization**: Individual brain scans are aligned to a common anatomical space, typically the MNI152 template. This step allows data from different subjects to be compared voxel-by-voxel or region-by-region in a standardized framework.
- **Smoothing and De-noising**: Spatial smoothing using a Gaussian kernel improves signal-to-noise ratio and accommodates anatomical variability. De-noising procedures such as regression of confounding signals (e.g., motion, white matter, CSF) help reduce artifacts and enhance data quality.
- **Temporal Filtering**: A band-pass filter is applied to the time-series data to retain frequencies typically associated with neural activity while removing low-frequency drift and high-frequency noise.
- **Parcellation with AAL Atlas**: The brain is divided into 116 distinct anatomical regions using the Automated Anatomical Labeling (AAL) atlas. For each region, the average BOLD signal across all voxels within that region is extracted, resulting in a region-wise time series.

#### Feature Engineering and Additional Steps

On top of the pre-processed time series, the following steps were performed in this study to prepare features suitable for machine learning:

- **Flattening the Correlation Matrix**: Since the Pearson correlation matrix is symmetric, only the upper triangular values (excluding the diagonal) are extracted and flattened into a 1D feature vector. This dramatically reduces the number of features while preserving all unique pairwise relationships.
- **Dimensionality Reduction**: To address the curse of dimensionality and enhance model generalization, feature reduction techniques such as Principal Component Analysis (PCA) or correlation-based selection can be applied. These helped remove redundant or non-informative features.

This carefully structured pre-processing pipeline ensures that the derived features are neuro-biologically grounded, noise-reduced, and optimized for machine learning-based classification between ASD and control subjects.

## 5.3 Feature Selection and Dimensionality Reduction

In the context of ML, high-dimensional datasets can lead to several challenges, including overfitting, increased computational load, and difficulty in model interpretation. This is especially true for the ABIDE dataset, which involves hundreds to thousands of functional connectivity features derived from correlation matrices, as well as the text-based dataset that includes multiple categorical and numerical attributes.

To address these issues, dimensionality reduction and feature selection techniques were employed. These approaches aim to retain the most relevant information while reducing redundancy, noise, and computational complexity.

### Principal Component Analysis (PCA)

Principal Component Analysis is a widely used linear technique for dimensionality reduction. In this work, PCA was largely used on the ABIDE dataset following feature extraction from the functional connectivity matrices. The key goals of using PCA were:

- **Noise Reduction**: By projecting the data onto directions of maximum variance, PCA helps eliminate features that contribute less to the overall variance (and often represent noise).
- **Redundancy Elimination**: Many brain areas show linked activity, resulting in redundant connectivity patterns. PCA combines linked features into orthogonal principal components.
- **Improved Training Efficiency**: Reducing the amount of features speeds up the training process and allows for the adoption of simpler models that perform better on untested data.
- **Visualization**: PCA also allows for data visualization in lower dimensions (e.g., 2D or 3D), helping assess cluster separation between ASD and control subjects.

The number of components maintained was determined using the cumulative explained variance criteria, which generally retains components that explain 95% or more of the overall variation.

### Rationale for Dimensionality Reduction

These feature selection and dimensionality reduction methods were crucial, particularly because:

- The **ABIDE dataset** involved 6670 unique pairwise correlations (from 116 AAL regions), which is large relative to the number of subjects.
- The **textual questionnaire data**, while not extremely high-dimensional, included categorical variables with multiple levels and required transformation to numeric space.

## 5.4 Tools and Technologies Used

This project's implementation required the use of a variety of robust open-source tools and libraries, the majority of which were created around the Python programming language. These tools helped at different phases of the workflow, including data pre-processing, model training, and visualization, allowing for effective handling of both structured questionnaire data and high-dimensional neuroimaging data.

### Python

Python formed the foundation of the entire project. Its rich ecosystem of scientific and machine learning libraries, paired with its ease of use and readability, made it perfect for quick prototyping, data analysis, and experimentation. Python's adaptability enabled the smooth integration of neuroimaging technologies with traditional machine learning frameworks.

### Pandas and NumPy

- **Pandas** was utilized for structured data processing, namely with questionnaire-based datasets. It was useful for dealing with missing values, encoding categorical characteristics, and converting tabular data into model-friendly representations.
- **NumPy** provided efficient array-based operations and numerical computations, which were essential during normalization, matrix transformations, and implementation of custom algorithms when needed.

### Scikit-learn

Scikit-learn was the primary machine learning framework used for both classification tasks and pre-processing. It provided built-in functions for:

- Splitting datasets into training and test sets
- Feature scaling and normalization
- Implementing classifiers such as Logistic Regression, SVM, and KNN
- Performing dimensionality reduction using PCA
- Evaluating model performance through accuracy, precision, recall, and F1-score metrics

Its intuitive API and well-documented utilities significantly accelerated development.

### Nilearn

Nilearn is a specialized Python library built on top of NiBabel and scikit-learn, designed for statistical learning on neuroimaging data. It played a crucial role in:

- Loading and handling 4D fMRI images from the ABIDE dataset
- Applying pre-processing steps such as masking and smoothing
- Extracting regional time series and computing functional connectivity matrices
- Reducing the complexity of raw neuroimaging data into structured features suitable for machine learning

This library was central to bridging the gap between raw brain imaging data and ML model inputs.

### Matplotlib and Seaborn

These visualization libraries were used for generating insightful plots during analysis. Their key use cases included:

- Plotting correlation heatmaps of features
- Visualizing the distribution of diagnostic labels
- Displaying performance metrics (e.g., confusion matrix, ROC curves)
- Tracking model behavior and feature relationships

Seaborn's high-level API helped in creating aesthetically appealing and informative statistical graphics.

### Jupyter Notebook and Visual Studio Code

- **Jupyter Notebook** was used to interactively explore, visualize, and record the data analysis process. The cell-based execution architecture allowed for gradual testing of pre-processing processes and algorithms.
- **Visual Studio Code (VS Code)** was used for organizing and managing larger scripts and modules, particularly when transitioning from prototype code to more structured implementations.

### GitHub

GitHub was used for version control, project tracking, and collaboration. The entire codebase, including data processing scripts, model implementations, and result visualizations, was maintained in a public repository. This ensured transparency, reproducibility, and ease of access for future research or peer review.

# 6. Machine Learning Models

## 6.1 For screening-questionnaire dataset:

### 1) Decision tree:

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the data based on feature values, creating a tree-like structure of decisions.

How It Works:

- The algorithm selects the best feature to split the data using criteria like Gini Impurity or Information Gain (entropy).
- This process continues recursively, creating branches until a stopping condition is met (e.g., maximum depth, pure leaves).

Key Concepts:

- Nodes: Represent features or questions.
- Branches: Represent decisions or outcomes from a node.
- Leaves: Represent final output values (classes or predicted numbers).

Block Diagram:

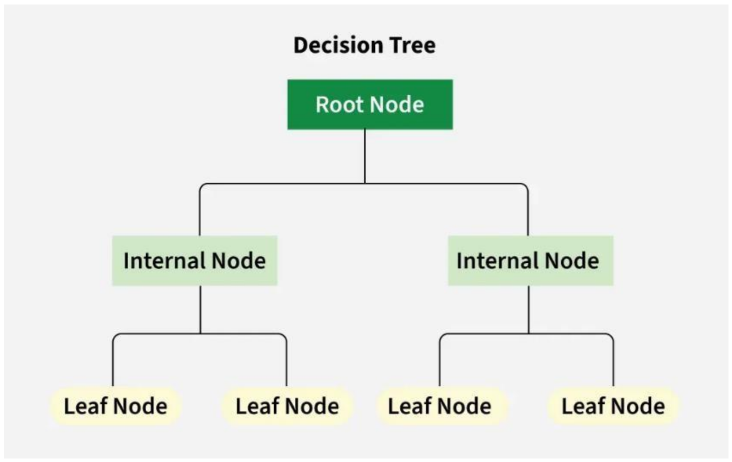


*Figure 1: Block diagram of Decision Tree Model (www.geeksforgeeks.com)*

Flowchart:

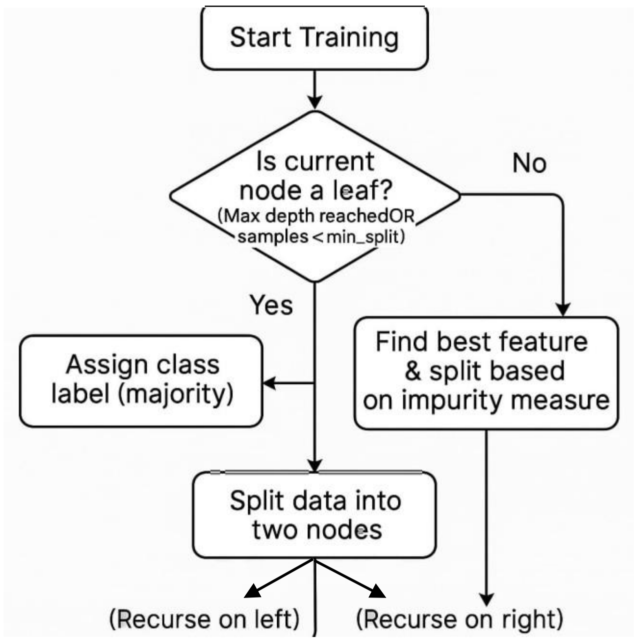

Start Training
Is current node a leaf? (Max depth reachedOR samples < min_split)
No
Yes
Assign class label (majority)
Find best feature & split based on impurity measure
Split data into two nodes
(Recurse on left)
(Recurse on right)

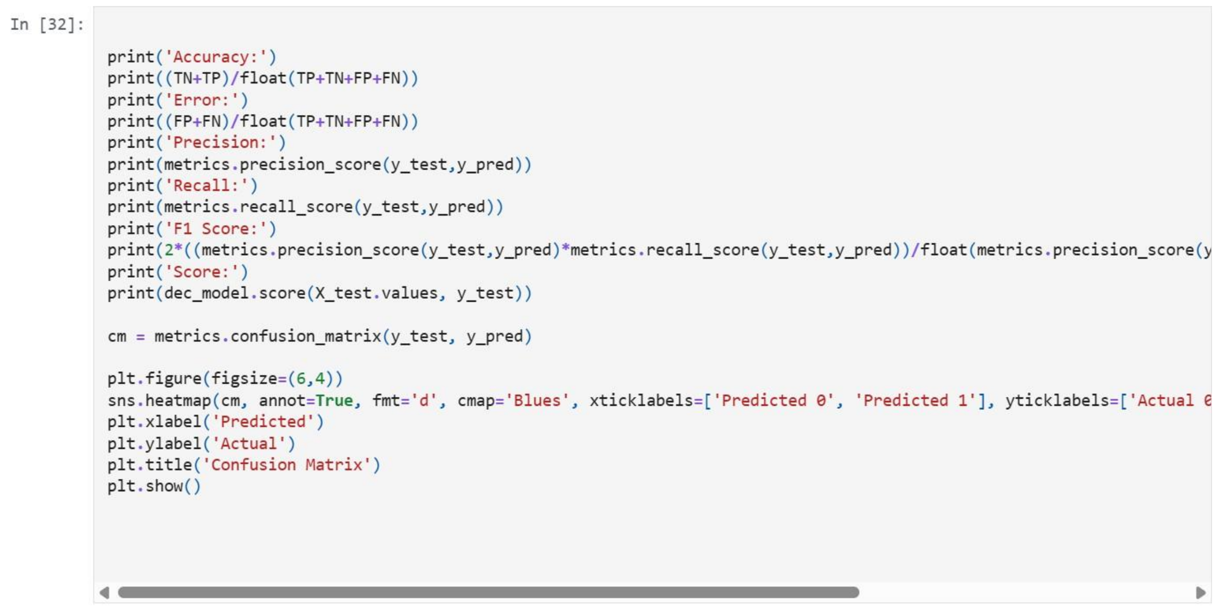
*Figure 2: Flowchart of Decision Tree Model*

Code snippet:

```
In [24]:

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
dec_model = DecisionTreeClassifier(max_depth=5, min_samples_split=10 , min_samples_leaf=2, max_features="sqrt")
dec_model.fit(X_train.values, y_train)

Out[24]: DecisionTreeClassifier(max_depth=5, max_features='sqrt', min_samples_leaf=2,
                                min_samples_split=10)
```

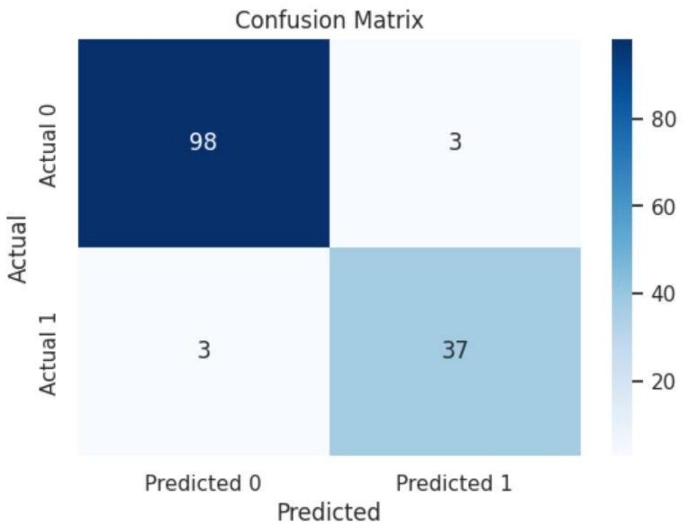
```
In [25]:

y_pred = dec_model.predict(X_test.values)
print('True : ', y_test.values[0:25])
print('False :', y_pred[0:25])

True :  [1 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0]
False : [1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0]
```

```
In [26]:

from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
TP = cm[1,1]
FP = cm[0,1]
TN = cm[0,0]
FN = cm[1,0]

[[98  3]
 [ 3 37]]
```



```
In [32]:

print('Accuracy:')
print((TN+TP)/float(TP+TN+FP+FN))
print('Error:')
print((FP+FN)/float(TP+TN+FP+FN))
print('Precision:')
print(metrics.precision_score(y_test,y_pred))
print('Recall:')
print(metrics.recall_score(y_test,y_pred))
print('F1 Score:')
print(2*((metrics.precision_score(y_test,y_pred)*metrics.recall_score(y_test,y_pred))/float(metrics.precision_score(y
print('Score:')
print(dec_model.score(X_test.values, y_test))

cm = metrics.confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy:
0.9574468085106383
Error:
0.0425531914893617
Precision:
0.925
Recall:
0.925
F1 Score:
0.925
Score:
0.9574468085106383
```



## 2) Random Forest:

A Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs to improve prediction accuracy and robustness. It is commonly used for both classification and regression problems.

How It Works:

- Creates many decision trees using different subsets of the training data (bootstrap sampling).

- At each split in a tree, only a random subset of features is considered, adding diversity.
- Final prediction is made by majority vote (classification) or average (regression) of all trees.

Key Features:

- Based on bagging (bootstrap aggregating).
- Reduces overfitting compared to a single decision tree.
- Increases accuracy and stability.

Flowchart:

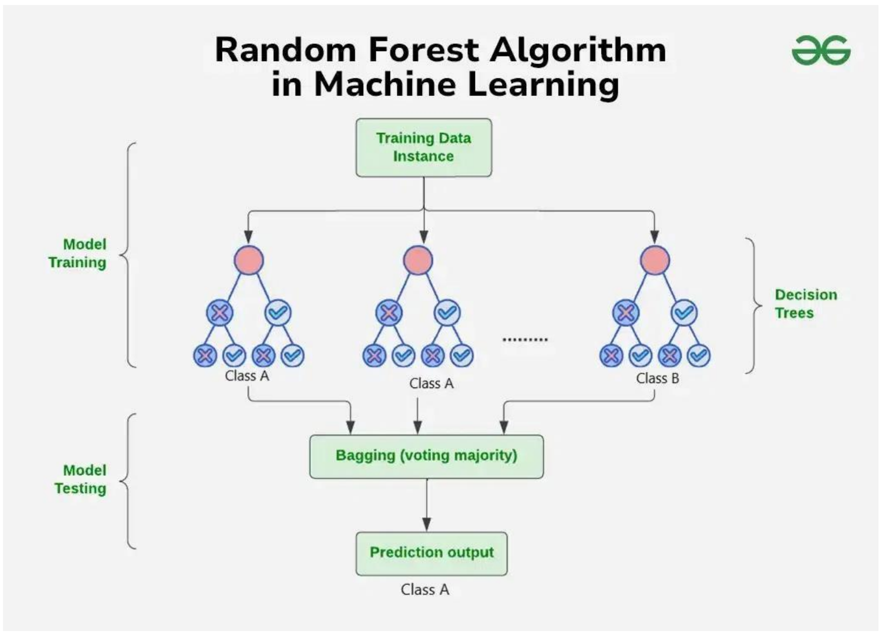
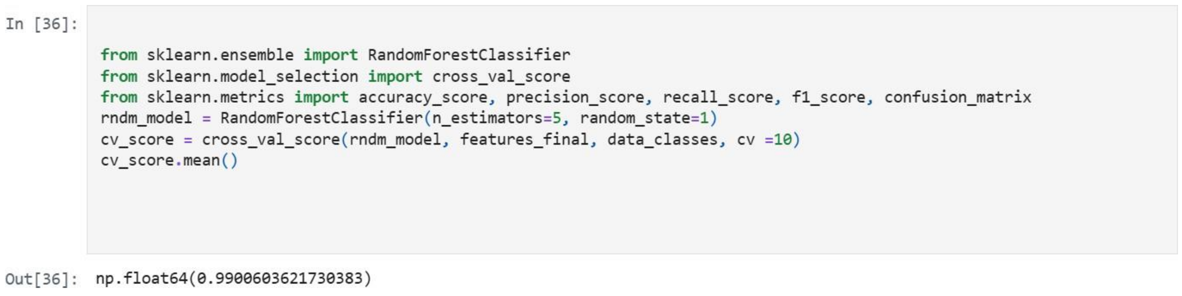
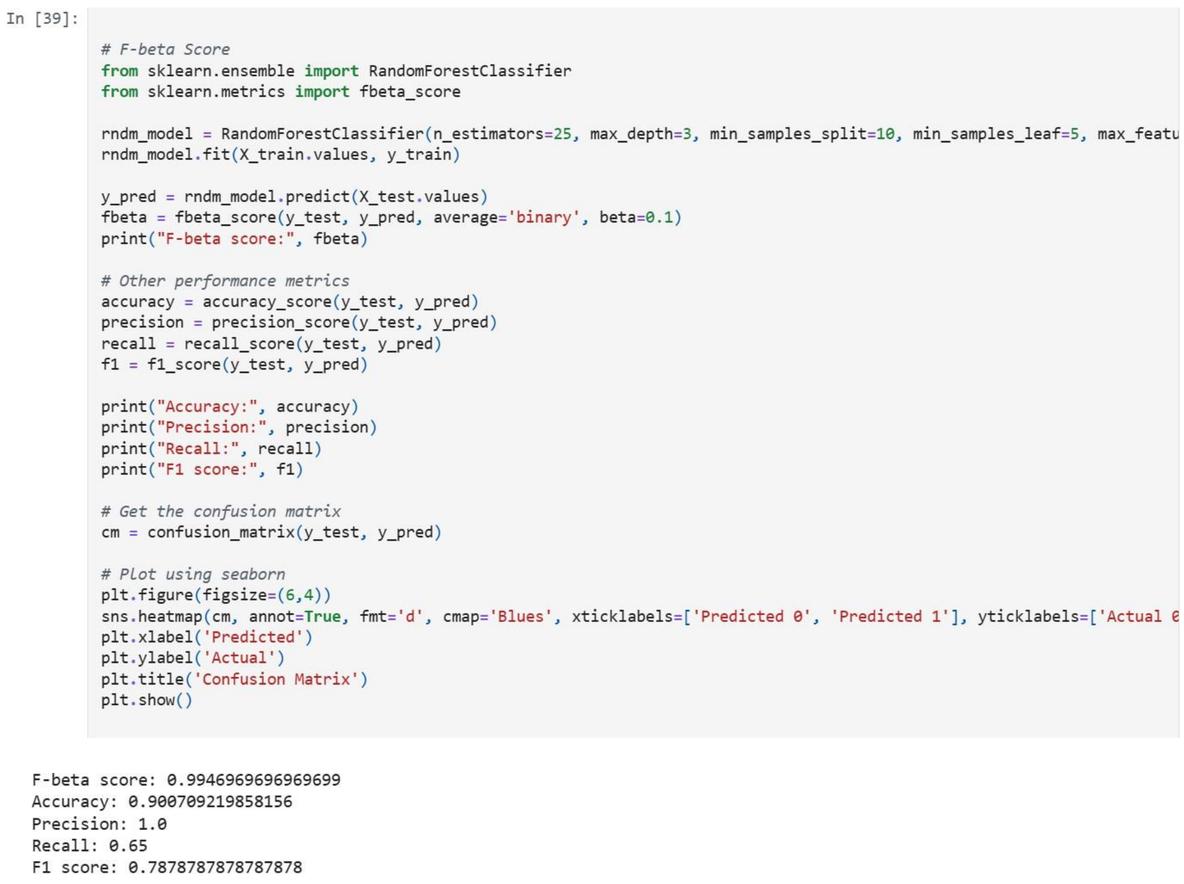


*Figure 3: Block Diagram of Random Forest Classifier (www.geeksforgeeks.com)*

Code snippet:

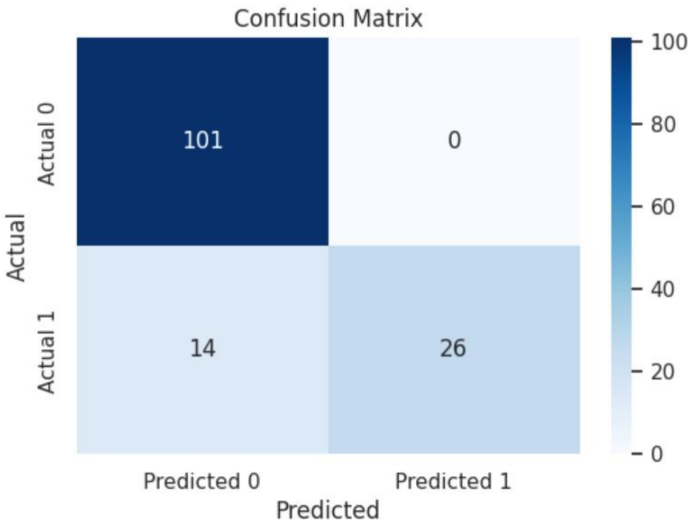


```
In [36]:
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
rndm_model = RandomForestClassifier(n_estimators=5, random_state=1)
cv_score = cross_val_score(rndm_model, features_final, data_classes, cv =10)
cv_score.mean()

Out[36]:  np.float64(0.9900603621730383)
```



```
In [39]:
         # F-beta Score
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import fbeta_score

         rndm_model = RandomForestClassifier(n_estimators=25, max_depth=3, min_samples_split=10, min_samples_leaf=5, max_featu
         rndm_model.fit(X_train.values, y_train)

         y_pred = rndm_model.predict(X_test.values)
         fbeta = fbeta_score(y_test, y_pred, average='binary', beta=0.1)
         print("F-beta score:", fbeta)

         # Other performance metrics
         accuracy = accuracy_score(y_test, y_pred)
         precision = precision_score(y_test, y_pred)
         recall = recall_score(y_test, y_pred)
         f1 = f1_score(y_test, y_pred)

         print("Accuracy:", accuracy)
         print("Precision:", precision)
         print("Recall:", recall)
         print("F1 score:", f1)

         # Get the confusion matrix
         cm = confusion_matrix(y_test, y_pred)

         # Plot using seaborn
         plt.figure(figsize=(6,4))
         sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.title('Confusion Matrix')
         plt.show()
```

```
F-beta score: 0.9946969696969699
Accuracy: 0.900709219858156
Precision: 1.0
Recall: 0.65
F1 score: 0.7878787878787878
```



### 3) Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised learning algorithm used primarily for classification, but also applicable to regression tasks. It aims to find the optimal hyperplane that best separates data points of different classes.

How It Works:

- SVM finds the hyperplane with the maximum margin, i.e., the greatest distance between the hyperplane and the nearest data points from each class (called support vectors).
- For non-linearly separable data, it uses a kernel function (e.g., RBF, polynomial) to transform the data into a higher-dimensional space where it becomes linearly separable.

Key Concepts:

- Support Vectors: Critical data points closest to the decision boundary.
- Hyperplane: The decision boundary that separates classes.
- Margin: Distance between support vectors and the hyperplane.

Flowchart:

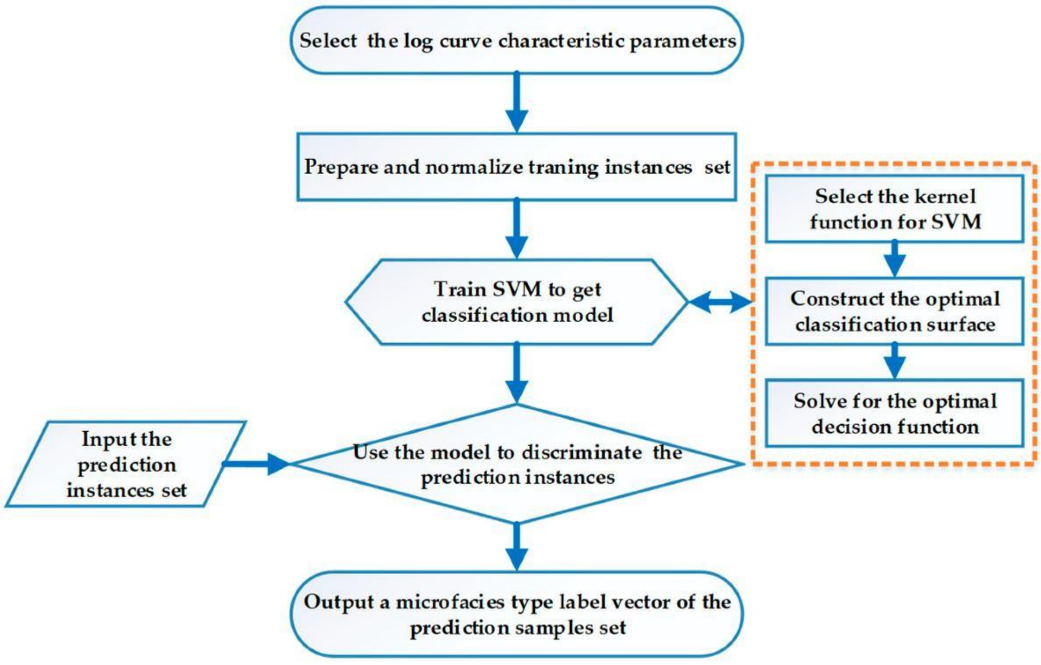
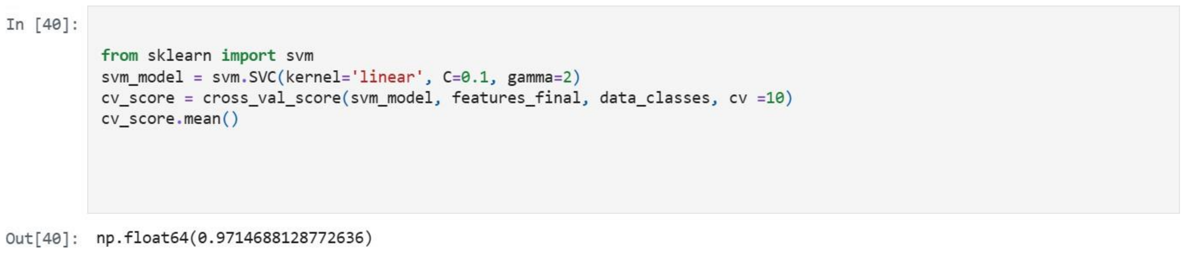
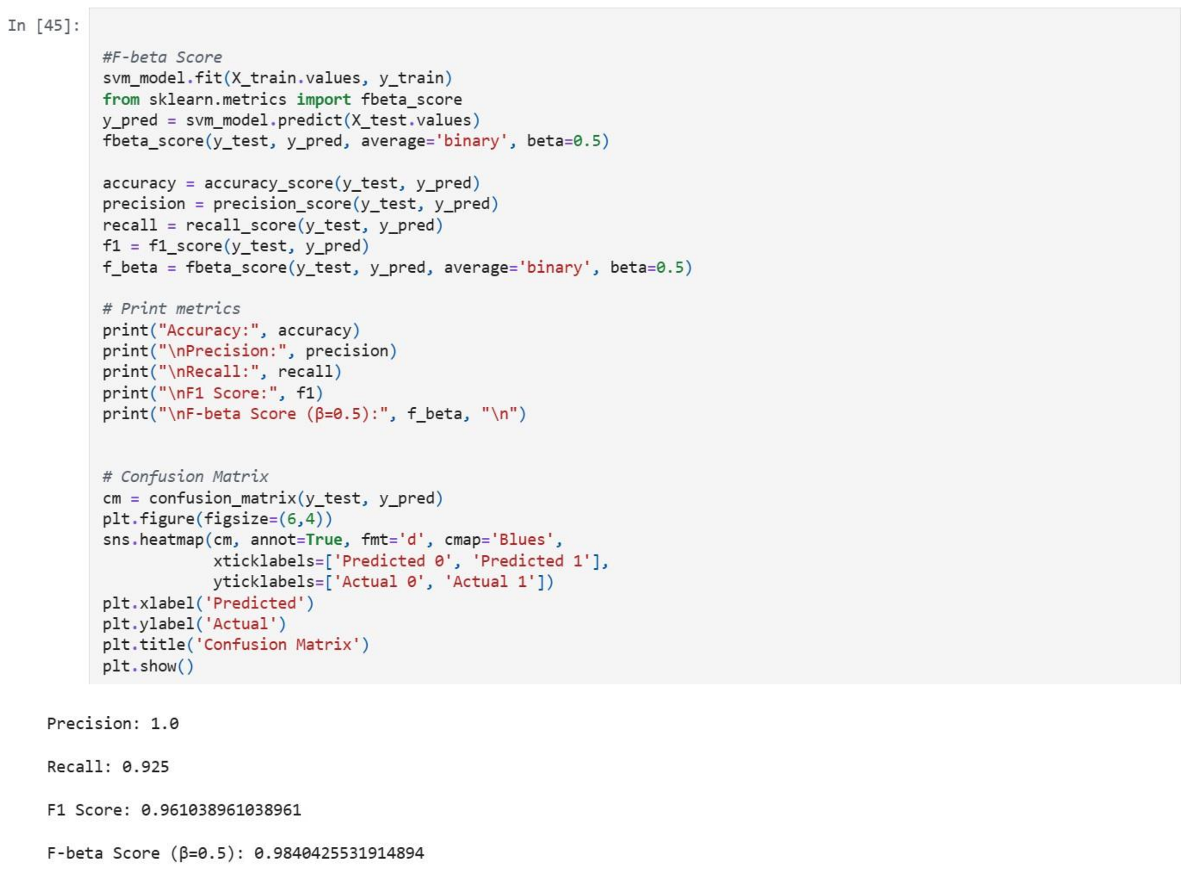

Select the log curve characteristic parameters
Prepare and normalize traning instances set
Select the kernel function for SVM
Train SVM to get classification model
Construct the optimal classification surface
Solve for the optimal decision function
Input the prediction instances set
Use the model to discriminate the prediction instances
Output a microfacies type label vector of the prediction samples set

*Figure 4: Flowchart for Support Vector Machines*

Code snippets:

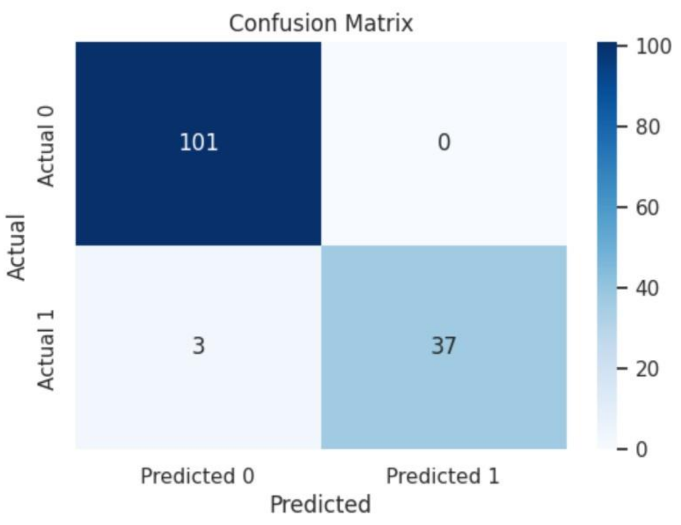


```
In [40]:
from sklearn import svm
svm_model = svm.SVC(kernel='linear', C=0.1, gamma=2)
cv_score = cross_val_score(svm_model, features_final, data_classes, cv =10)
cv_score.mean()

Out[40]: np.float64(0.9714688128772636)
```



```
In [45]:
#F-beta Score
svm_model.fit(X_train.values, y_train)
from sklearn.metrics import fbeta_score
y_pred = svm_model.predict(X_test.values)
fbeta_score(y_test, y_pred, average='binary', beta=0.5)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
f_beta = fbeta_score(y_test, y_pred, average='binary', beta=0.5)

# Print metrics
print("Accuracy:", accuracy)
print("\nPrecision:", precision)
print("\nRecall:", recall)
print("\nF1 Score:", f1)
print("\nF-beta Score (β=0.5):", f_beta, "\n")


# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Precision: 1.0

Recall: 0.925

F1 Score: 0.961038961038961

F-beta Score (β=0.5): 0.9840425531914894
```



### 4) K-Nearest-Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple, instance-based, supervised machine learning algorithm used for classification and regression tasks. It makes predictions based on the 'k' closest training samples in the feature space.

How It Works:

- For a given input, the algorithm calculates the distance (usually Euclidean) to all training points.
- It selects the 'k' nearest neighbors.
- For classification, it predicts the majority class among the neighbors.
- For regression, it returns the average of the neighbors' values.

Key Concepts:

- Distance Metrics: Commonly Euclidean, Manhattan, or Minkowski distance.
- Value of 'k': A small 'k' may overfit; a large 'k' may underfit.
- Lazy Learning: No actual training phase; computation happens during prediction.

Flowchart:

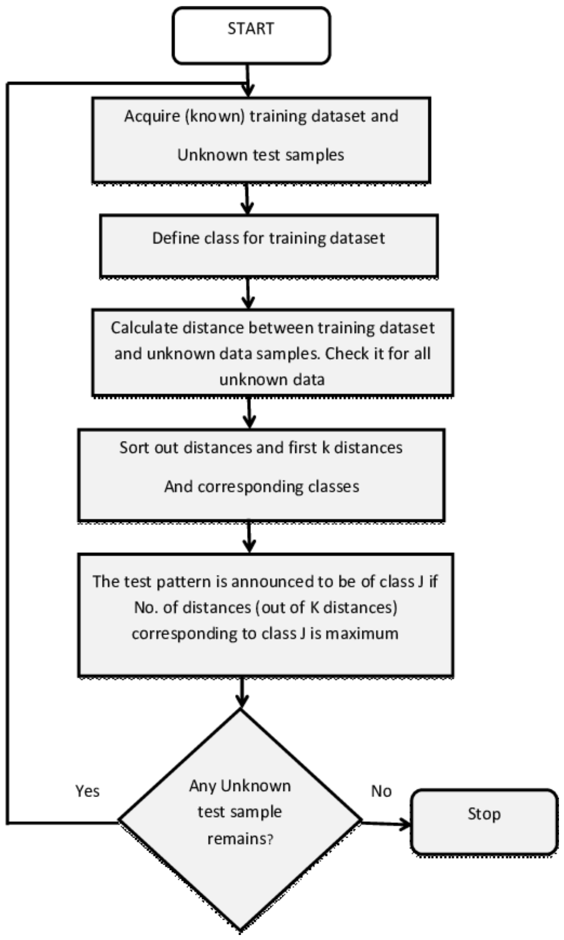

START
Acquire (known) training dataset and Unknown test samples
Define class for training dataset
Calculate distance between training dataset and unknown data samples. Check it for all unknown data
Sort out distances and first k distances And corresponding classes
The test pattern is announced to be of class J if No. of distances (out of K distances) corresponding to class J is maximum
Any Unknown test sample remains?
Yes
No
Stop

*Figure 5: Flowchart for K Nearest Neighbors*

Code Snippet:

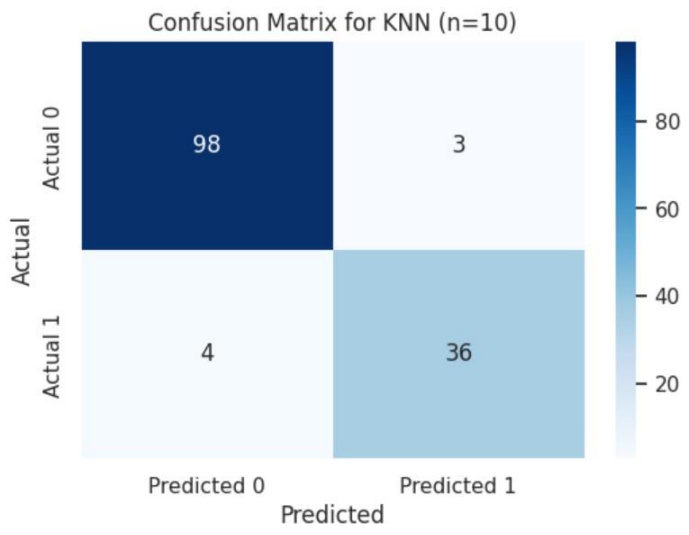
```
In [46]:
from sklearn import neighbors
knn_model = neighbors.KNeighborsClassifier(n_neighbors=10)
cv_score = cross_val_score(knn_model, features_final, data_classes, cv =10)
cv_score.mean()
```

```
Out[46]: np.float64(0.9458752515090543)
```

```
In [47]:
#F-beta Score
knn_model.fit(X_train.values, y_train)
from sklearn.metrics import fbeta_score
y_pred = knn_model.predict(X_test.values)
fbeta = fbeta_score(y_test, y_pred, average='binary', beta=0.5)
```

```
In [48]:
# Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
f_beta = fbeta_score(y_test, y_pred, average='binary', beta=0.5)

print("\nEvaluation Metrics for KNN (n=10):")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("F-beta Score (β=0.5):", f_beta)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for KNN (n=10)')
plt.show()
```

```
Evaluation Metrics for KNN (n=10):
Accuracy: 0.950354609929078
Precision: 0.9230769230769231
Recall: 0.9
F1 Score: 0.9113924050632911
F-beta Score (β=0.5): 0.9183673469387755
```



## 5) Naïve Bayes

Naive Bayes is a family of probabilistic classifiers based on Bayes' Theorem, with the assumption of feature independence. It is widely used for classification tasks and is especially effective for text data.

How It Works:

- Uses Bayes' Theorem to calculate the posterior probability of each class given the input features.
- Assumes that all features contribute independently to the outcome (hence "naive").
- Chooses the class with the highest posterior probability.

Bayes' Theorem:

$$P(C \mid X) = \frac{P(X \mid C) \cdot P(C)}{P(X)}$$

Where:

- P(C|X): Posterior probability of class CCC given features XXX
- P(X|C)): Likelihood of features given class
- P(C): Prior probability of class
- P(X): Prior probability of features (can be ignored in classification)

Common Variants:

- Gaussian Naive Bayes: Assumes features follow a normal distribution.
- Multinomial Naive Bayes: Ideal for word counts (e.g., text classification).
- Bernoulli Naive Bayes: For binary/boolean features.

Flowchart:

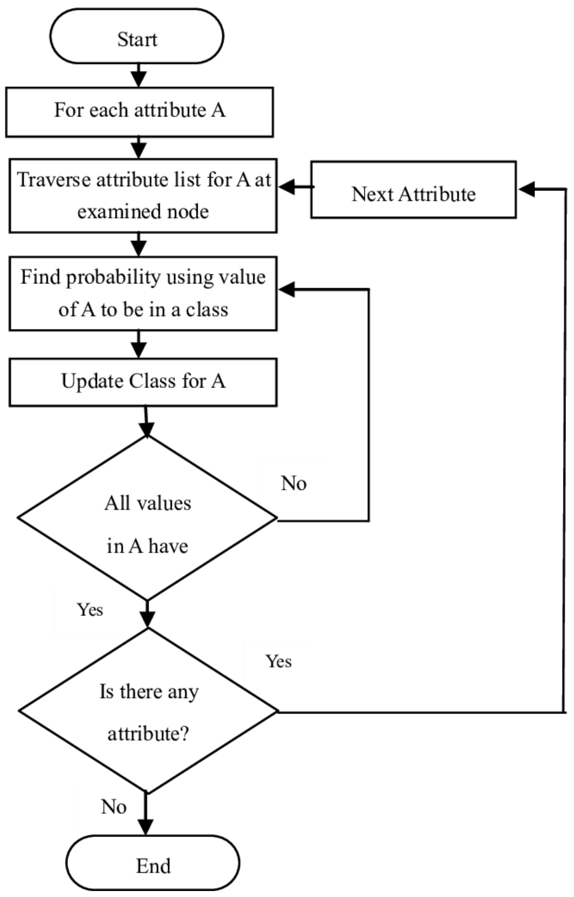
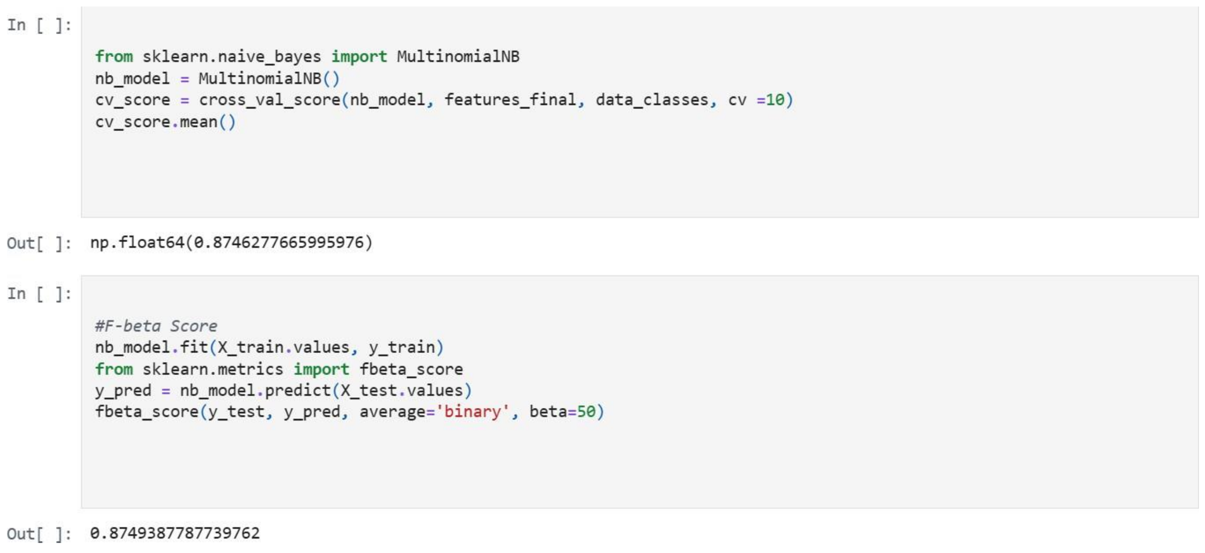


*Figure 6: Flowchart for Naive Bayes Classifier*

Code Snippet:

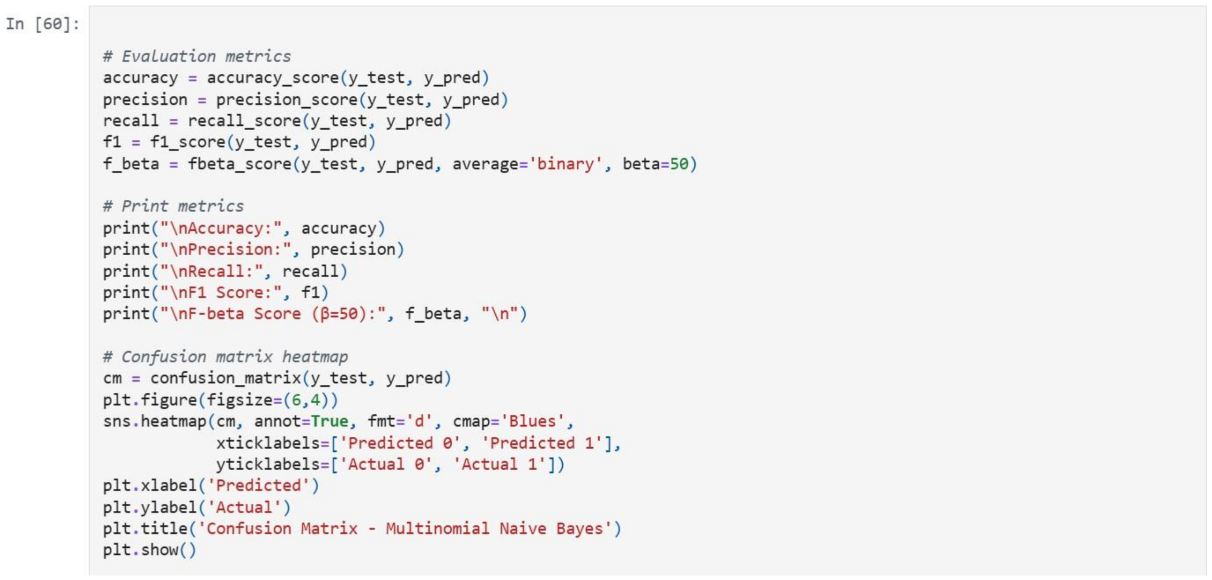


```
In [ ]:
from sklearn.naive_bayes import MultinomialNB
nb_model = MultinomialNB()
cv_score = cross_val_score(nb_model, features_final, data_classes, cv =10)
cv_score.mean()
```

```
Out[ ]: np.float64(0.8746277665995976)
```

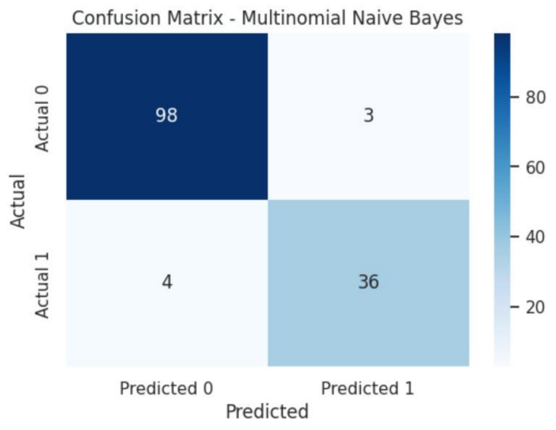
```
In [ ]:
#F-beta Score
nb_model.fit(X_train.values, y_train)
from sklearn.metrics import fbeta_score
y_pred = nb_model.predict(X_test.values)
fbeta_score(y_test, y_pred, average='binary', beta=50)
```

```
Out[ ]: 0.8749387787739762
```



```
In [60]:
# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
f_beta = fbeta_score(y_test, y_pred, average='binary', beta=50)

# Print metrics
print("\nAccuracy:", accuracy)
print("\nPrecision:", precision)
print("\nRecall:", recall)
print("\nF1 Score:", f1)
print("\nF-beta Score (β=50):", f_beta, "\n")

# Confusion matrix heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Multinomial Naive Bayes')
plt.show()
```

```
Accuracy: 0.950354609929078

Precision: 0.9230769230769231

Recall: 0.9

F1 Score: 0.9113924050632911

F-beta Score (β=50): 0.9000089964913683
```



### 6) Logistic Regression

Logistic Regression is a supervised learning algorithm used for binary and multiclass classification problems. Despite its name, it is a classification model—not a regression one—and predicts the probability that a given input belongs to a particular class.

How It Works:

- Uses a logistic (sigmoid) function to map predicted values to probabilities between 0 and 1.
- Calculates a linear combination of input features, then applies the sigmoid function:

$$P(y = 1 \mid X) = \frac{1}{1 + e - (\beta 0 + \beta 1X1 + \beta 2X2 + \cdots + \beta nXn)}$$

- If the probability ≥ 0.5, it predicts class 1; otherwise, class 0.

Key Concepts:

- Outputs a probability score rather than a hard label.
- Can be extended to multiclass classification using methods like one-vs-rest (OvR) or softmax regression.

Flowchart:

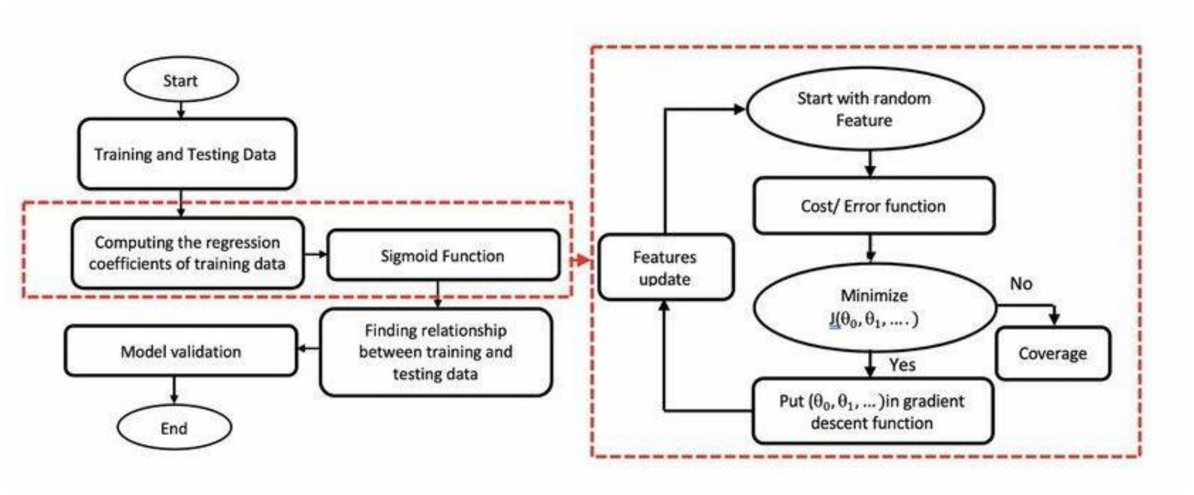
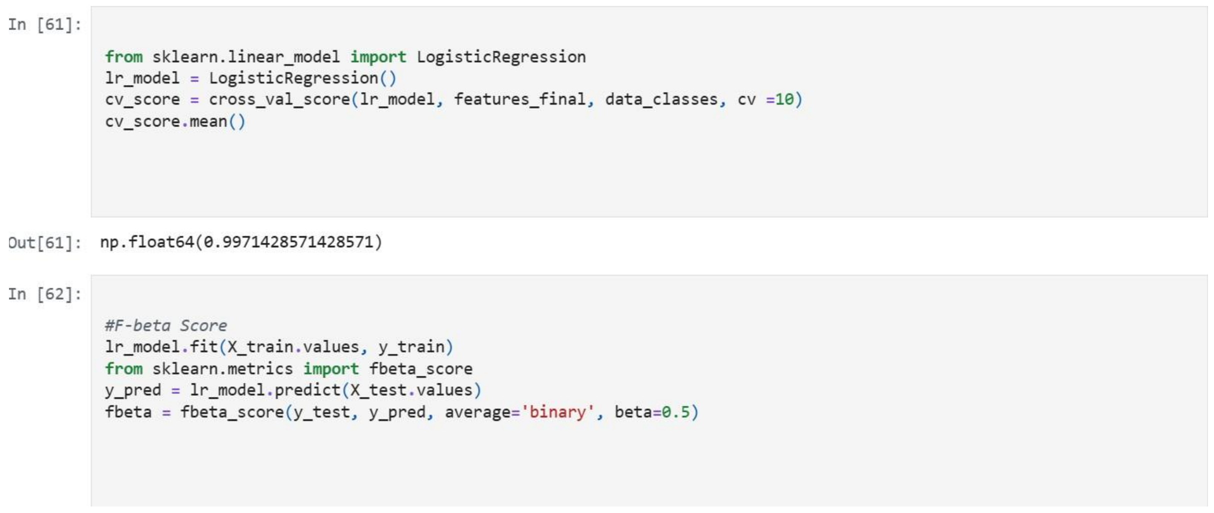
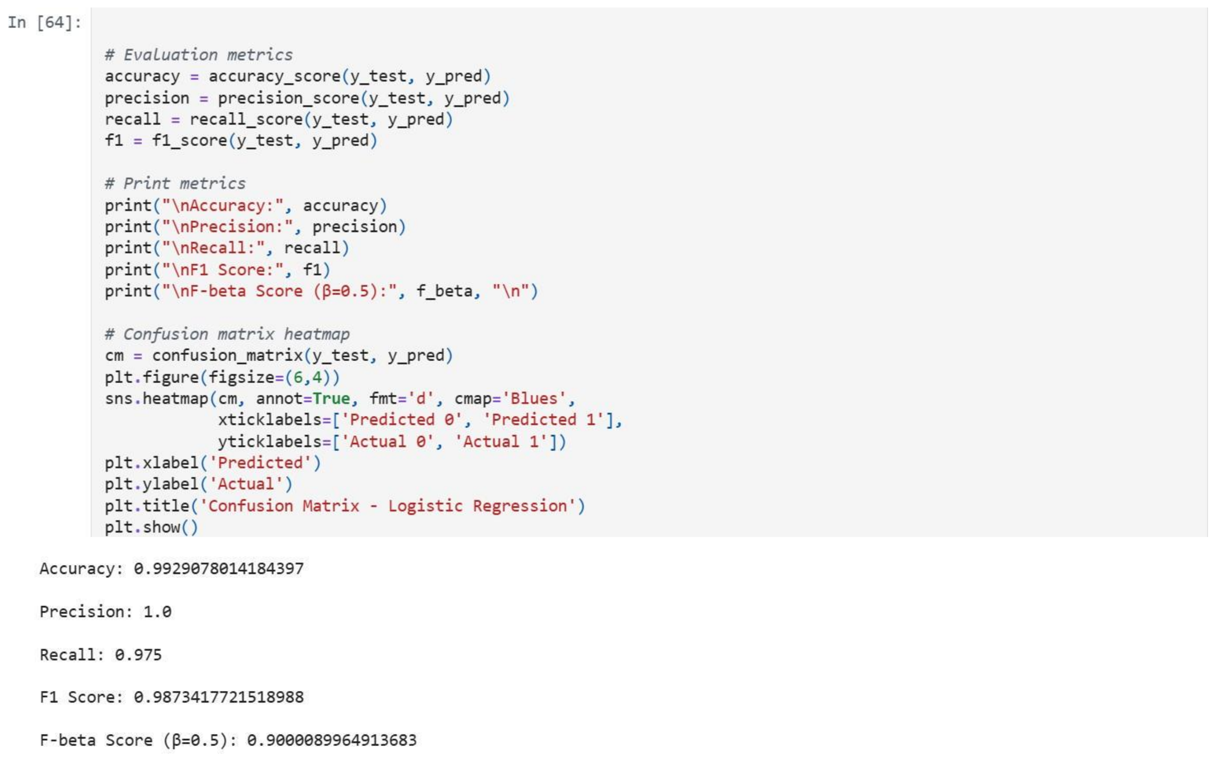

Start
Training and Testing Data
Computing the regression coefficients of training data
Sigmoid Function
Finding relationship between training and testing data
Model validation
End
Features update
Start with random Feature
Cost/ Error function
Minimize J(θ0, θ1, ....)
No
Coverage
Yes
Put (θ0, θ1, ...)in gradient descent function

*Figure 7: Flowchart for Logistic Regression*

Code Snippet:

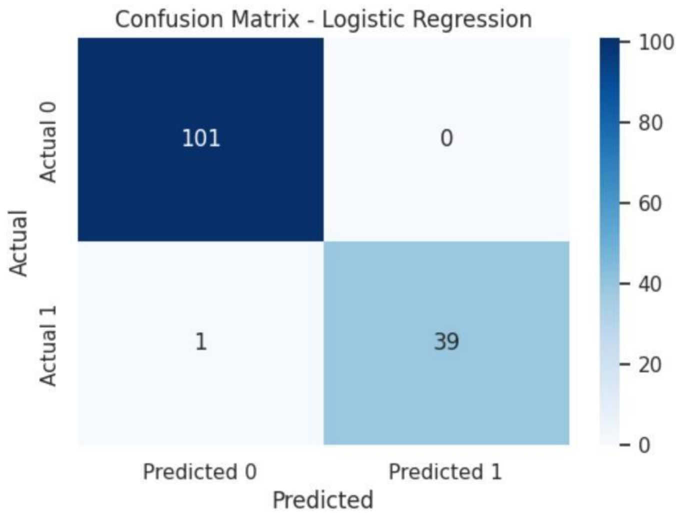


```
In [61]:
    from sklearn.linear_model import LogisticRegression
    lr_model = LogisticRegression()
    cv_score = cross_val_score(lr_model, features_final, data_classes, cv =10)
    cv_score.mean()

Out[61]: np.float64(0.9971428571428571)

In [62]:
    #F-beta Score
    lr_model.fit(X_train.values, y_train)
    from sklearn.metrics import fbeta_score
    y_pred = lr_model.predict(X_test.values)
    fbeta = fbeta_score(y_test, y_pred, average='binary', beta=0.5)
```



```
In [64]:
         # Evaluation metrics
         accuracy = accuracy_score(y_test, y_pred)
         precision = precision_score(y_test, y_pred)
         recall = recall_score(y_test, y_pred)
         f1 = f1_score(y_test, y_pred)

         # Print metrics
         print("\nAccuracy:", accuracy)
         print("\nPrecision:", precision)
         print("\nRecall:", recall)
         print("\nF1 Score:", f1)
         print("\nF-beta Score (β=0.5):", f_beta, "\n")

         # Confusion matrix heatmap
         cm = confusion_matrix(y_test, y_pred)
         plt.figure(figsize=(6,4))
         sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                     xticklabels=['Predicted 0', 'Predicted 1'],
                     yticklabels=['Actual 0', 'Actual 1'])
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.title('Confusion Matrix - Logistic Regression')
         plt.show()
```

```
Accuracy: 0.9929078014184397

Precision: 1.0

Recall: 0.975

F1 Score: 0.9873417721518988

F-beta Score (β=0.5): 0.9000089964913683
```



## 6.2 For ABIDE dataset

### 1) K-Nearest Neighbor (KNN)

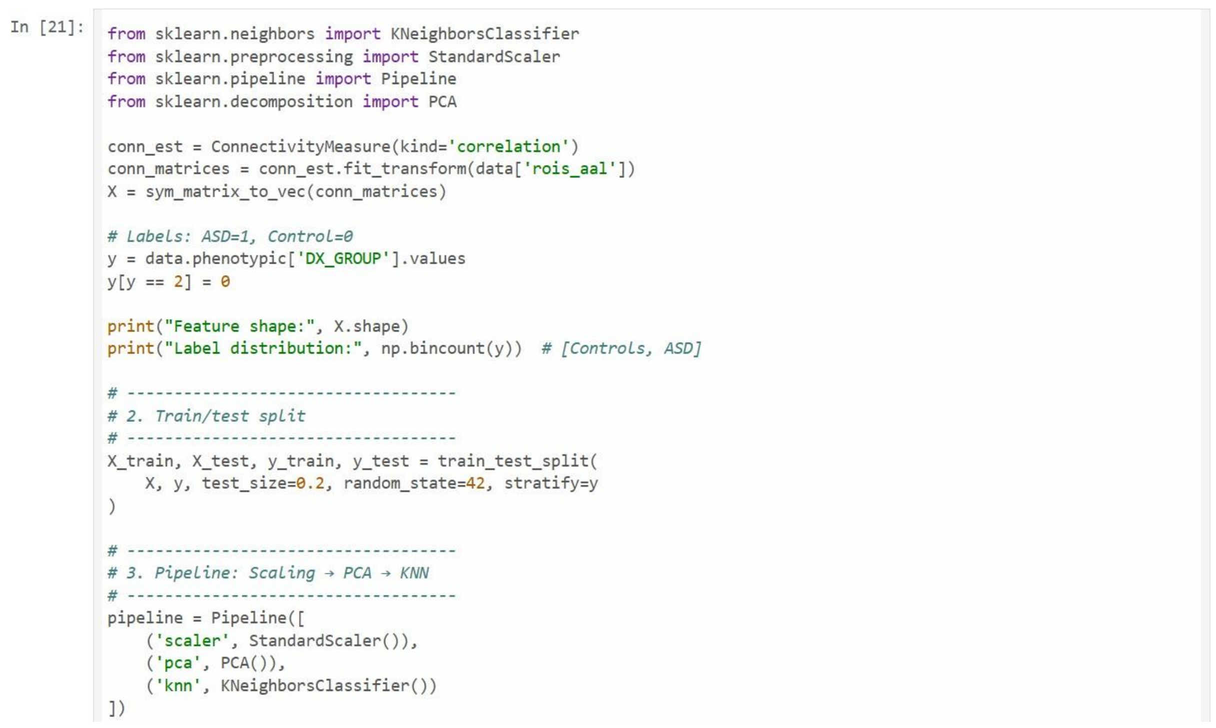
K-Nearest Neighbors (KNN) is a simple, instance-based, supervised machine learning algorithm used for classification and regression tasks. It makes predictions based on the 'k' closest training samples in the feature space.

How It Works:

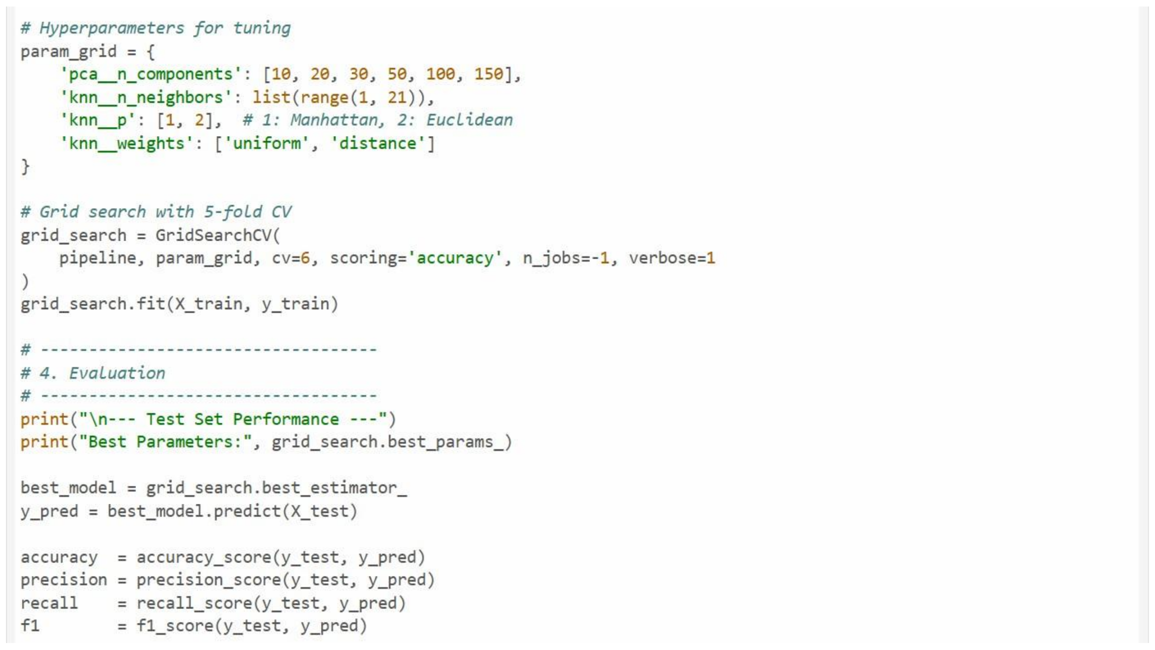
- For a given input, the algorithm calculates the distance (usually Euclidean) to all training points.
- It selects the 'k' nearest neighbors.

- For classification, it predicts the majority class among the neighbors.
- For regression, it returns the average of the neighbors' values.

Code snippet:



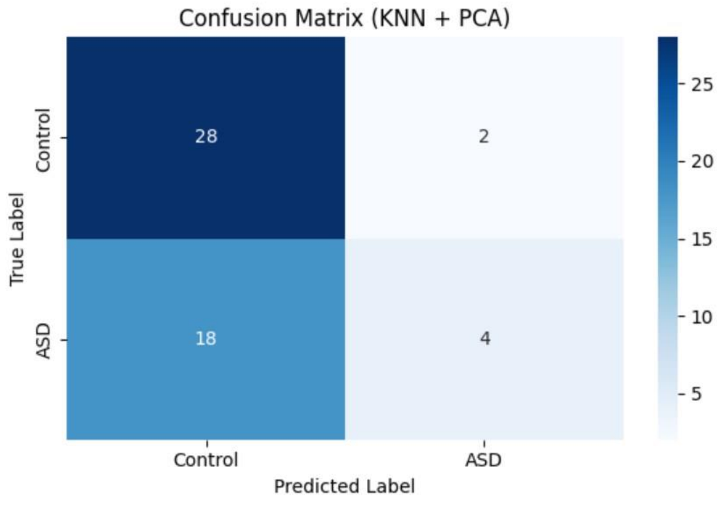
```
In [21]:  from sklearn.neighbors import KNeighborsClassifier
          from sklearn.preprocessing import StandardScaler
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import PCA

          conn_est = ConnectivityMeasure(kind='correlation')
          conn_matrices = conn_est.fit_transform(data['rois_aal'])
          X = sym_matrix_to_vec(conn_matrices)

          # Labels: ASD=1, Control=0
          y = data.phenotypic['DX_GROUP'].values
          y[y == 2] = 0

          print("Feature shape:", X.shape)
          print("Label distribution:", np.bincount(y))  # [Controls, ASD]

          # ----------------------------------
          # 2. Train/test split
          # ----------------------------------
          X_train, X_test, y_train, y_test = train_test_split(
              X, y, test_size=0.2, random_state=42, stratify=y
          )

          # ----------------------------------
          # 3. Pipeline: Scaling → PCA → KNN
          # ----------------------------------
          pipeline = Pipeline([
              ('scaler', StandardScaler()),
              ('pca', PCA()),
              ('knn', KNeighborsClassifier())
          ])
```



```
# Hyperparameters for tuning
param_grid = {
    'pca__n_components': [10, 20, 30, 50, 100, 150],
    'knn__n_neighbors': list(range(1, 21)),
    'knn__p': [1, 2],  # 1: Manhattan, 2: Euclidean
    'knn__weights': ['uniform', 'distance']
}

# Grid search with 5-fold CV
grid_search = GridSearchCV(
    pipeline, param_grid, cv=6, scoring='accuracy', n_jobs=-1, verbose=1
)
grid_search.fit(X_train, y_train)

# ----------------------------------
# 4. Evaluation
# ----------------------------------
print("\n--- Test Set Performance ---")
print("Best Parameters:", grid_search.best_params_)

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

accuracy  = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall    = recall_score(y_test, y_pred)
f1        = f1_score(y_test, y_pred)
```

```
print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1 Score:  {f1:.4f}")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Control', 'ASD'], yticklabels=['Control', 'ASD'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix (KNN + PCA)')
plt.tight_layout()
plt.show()

# -----------------------------------
# 5. Cross-Validation Accuracy
# -----------------------------------
cv_results = cross_validate(best_model, X, y, cv=5, scoring='accuracy')
cv_acc = np.mean(cv_results['test_score'])
print(f"\nCross-Validation Accuracy (entire dataset): {cv_acc:.4f}")
```

```
Feature shape: (258, 6786)
Label distribution: [150 108]
Fitting 6 folds for each of 480 candidates, totalling 2880 fits

--- Test Set Performance ---
Best Parameters: {'knn__n_neighbors': 17, 'knn__p': 1, 'knn__weights': 'distance', 'pca__n_components': 100}
Accuracy:  0.6154
Precision: 0.6667
Recall:    0.1818
F1 Score:  0.2857
```



```
Cross-Validation Accuracy (entire dataset): 0.6126
```

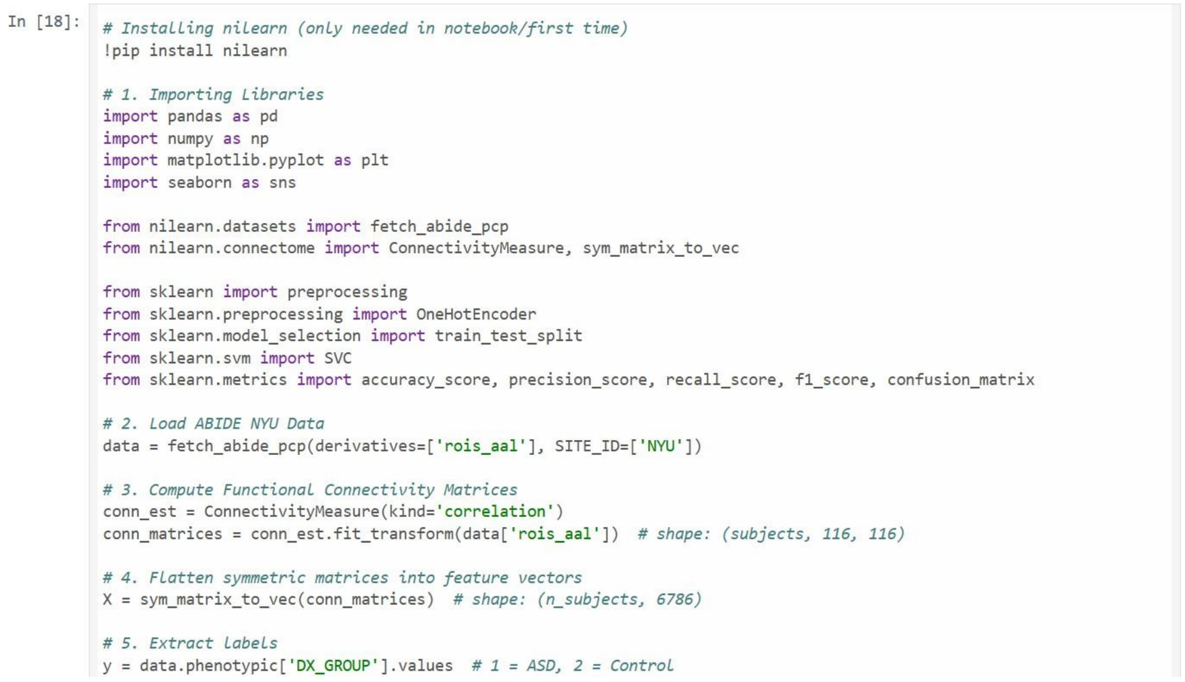
### 2) Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used primarily for classification, but also applicable to regression tasks. It aims to find the optimal hyperplane that best separates data points of different classes.

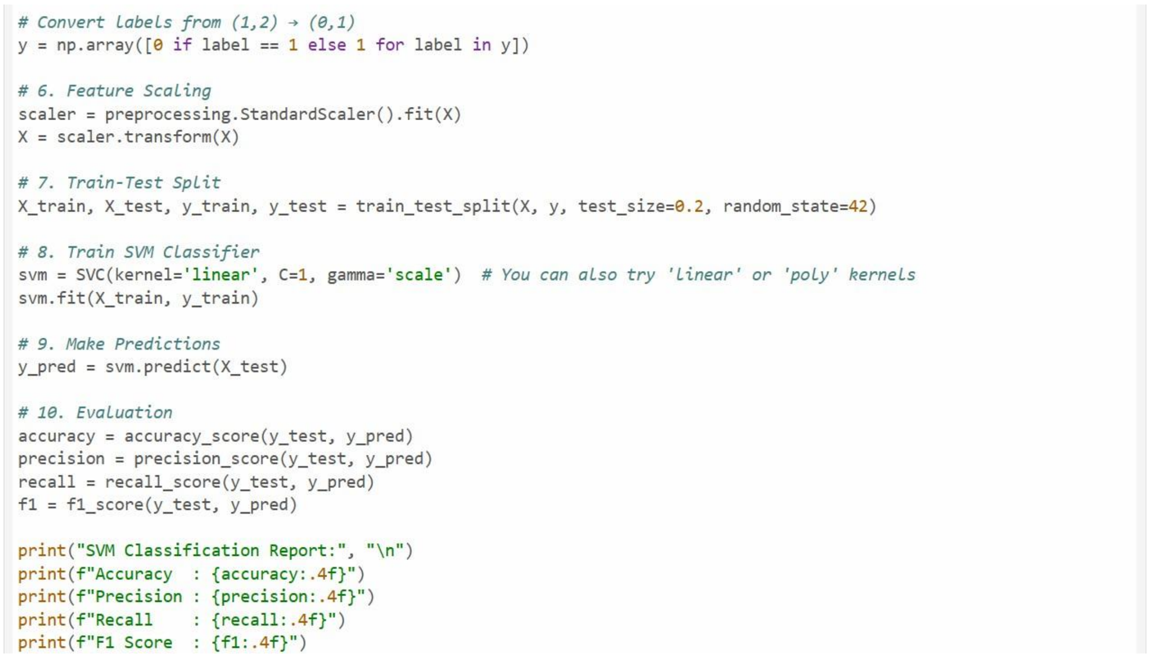
How It Works:

- SVM finds the hyperplane with the maximum margin, i.e., the greatest distance between the hyperplane and the nearest data points from each class (called support vectors).
- For non-linearly separable data, it uses a kernel function (e.g., RBF, polynomial) to transform the data into a higher-dimensional space where it becomes linearly separable.

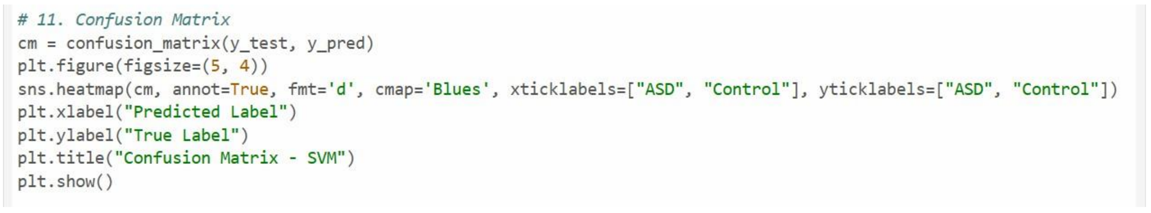
Code snippet:



```
In [18]:  # Installing nilearn (only needed in notebook/first time)
          !pip install nilearn

          # 1. Importing Libraries
          import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns

          from nilearn.datasets import fetch_abide_pcp
          from nilearn.connectome import ConnectivityMeasure, sym_matrix_to_vec

          from sklearn import preprocessing
          from sklearn.preprocessing import OneHotEncoder
          from sklearn.model_selection import train_test_split
          from sklearn.svm import SVC
          from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

          # 2. Load ABIDE NYU Data
          data = fetch_abide_pcp(derivatives=['rois_aal'], SITE_ID=['NYU'])

          # 3. Compute Functional Connectivity Matrices
          conn_est = ConnectivityMeasure(kind='correlation')
          conn_matrices = conn_est.fit_transform(data['rois_aal'])  # shape: (subjects, 116, 116)

          # 4. Flatten symmetric matrices into feature vectors
          X = sym_matrix_to_vec(conn_matrices)  # shape: (n_subjects, 6786)

          # 5. Extract labels
          y = data.phenotypic['DX_GROUP'].values  # 1 = ASD, 2 = Control
```

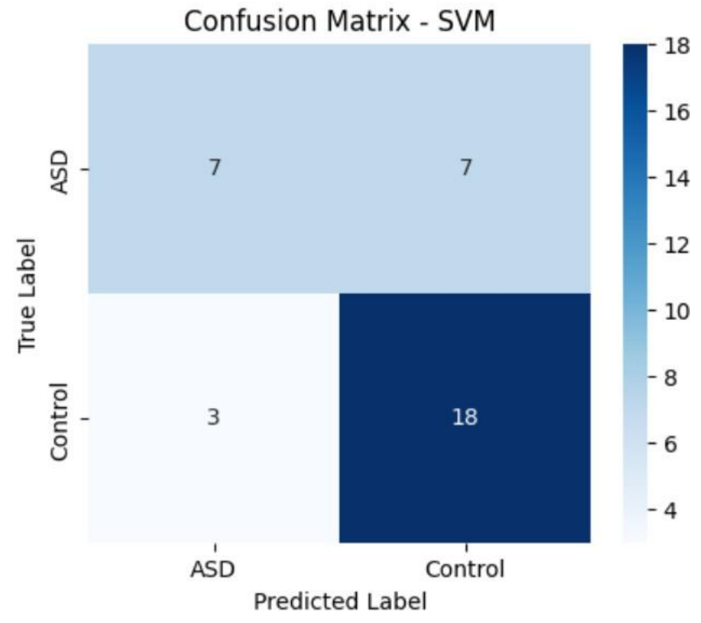


```
# Convert labels from (1,2) → (0,1)
y = np.array([0 if label == 1 else 1 for label in y])

# 6. Feature Scaling
scaler = preprocessing.StandardScaler().fit(X)
X = scaler.transform(X)

# 7. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 8. Train SVM Classifier
svm = SVC(kernel='linear', C=1, gamma='scale')  # You can also try 'linear' or 'poly' kernels
svm.fit(X_train, y_train)

# 9. Make Predictions
y_pred = svm.predict(X_test)

# 10. Evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("SVM Classification Report:", "\n")
print(f"Accuracy  : {accuracy:.4f}")
print(f"Precision : {precision:.4f}")
print(f"Recall    : {recall:.4f}")
print(f"F1 Score  : {f1:.4f}")
```



```
# 11. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["ASD", "Control"], yticklabels=["ASD", "Control"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - SVM")
plt.show()
```

```
SVM Classification Report:

Accuracy  : 0.7143
Precision : 0.7200
Recall    : 0.8571
F1 Score  : 0.7826
```



### 3) Logistic Regression

Logistic Regression is a supervised learning algorithm used for binary and multiclass classification problems. Despite its name, it is a classification model—not a regression one—and predicts the probability that a given input belongs to a particular class.

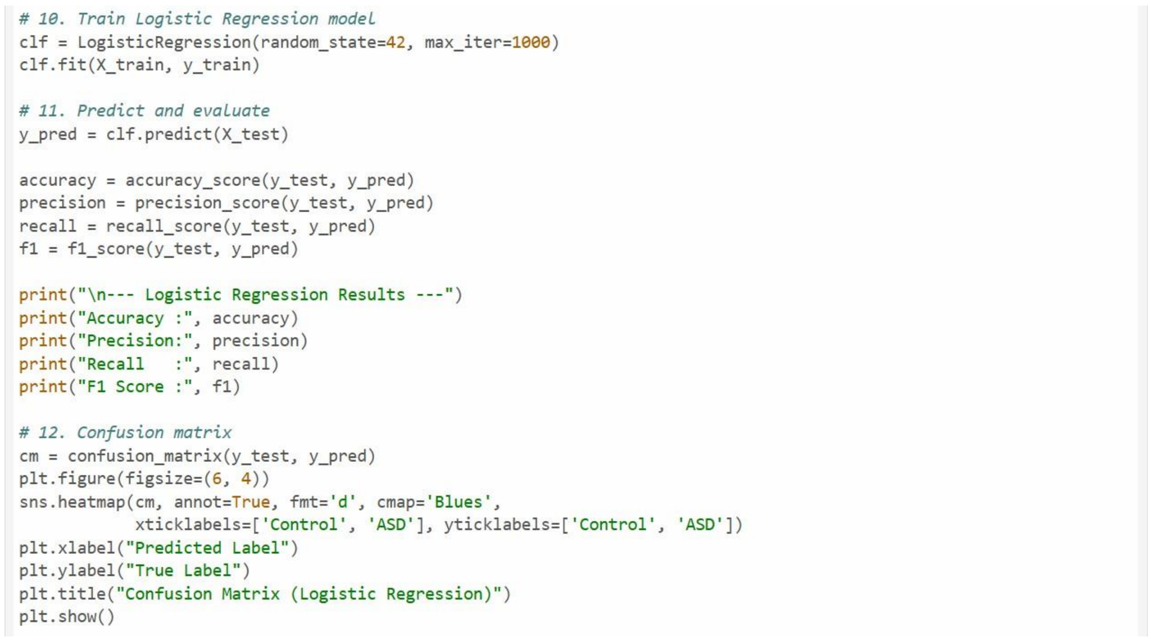
How It Works:

- Uses a logistic (sigmoid) function to map predicted values to probabilities between 0 and 1.
- Calculates a linear combination of input features, then applies the sigmoid function:

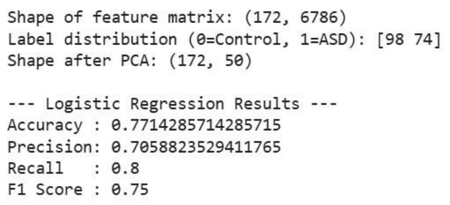
$$P(y = 1 \mid X) = \frac{1}{1 + e - (\beta 0 + \beta 1X1 + \beta 2X2 + \cdots + \beta nXn)}$$

- If the probability ≥ 0.5, it predicts class 1; otherwise, class 0.

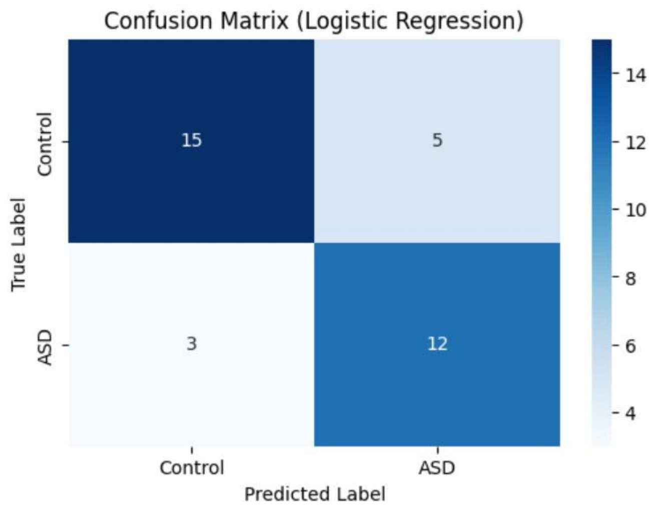
Code snippet:



```python
# 10. Train Logistic Regression model
clf = LogisticRegression(random_state=42, max_iter=1000)
clf.fit(X_train, y_train)

# 11. Predict and evaluate
y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("\n--- Logistic Regression Results ---")
print("Accuracy :", accuracy)
print("Precision:", precision)
print("Recall   :", recall)
print("F1 Score :", f1)

# 12. Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Control', 'ASD'], yticklabels=['Control', 'ASD'])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix (Logistic Regression)")
plt.show()
```



```
Shape of feature matrix: (172, 6786)
Label distribution (0=Control, 1=ASD): [98 74]
Shape after PCA: (172, 50)

--- Logistic Regression Results ---
Accuracy : 0.7714285714285715
Precision: 0.7058823529411765
Recall   : 0.8
F1 Score : 0.75
```



# 7. Experimental Results and Analysis

## 7.1 Evaluation Metrics

Evaluation metrics are used to assess the performance of trained models. We employed the following evaluation metrics on our models.

i. **Accuracy**: Accuracy measures the ratio of correct predictions made by the model to the total number of predictions. It gives us a general idea of how often the model is correct, but can be unreliable for unbalanced datasets.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

ii. **Precision**: Precision measures the accuracy of positive predictions. It is the ratio of true positive predictions to the total number of positive predictions that was made by the model. Models having a high precision have a very low rate false positives.

$$Precision = \frac{TP}{TP + FP}$$

iii. **Recall (or Sensitivity)**: Recall measures the ability of a model to correctly identify all relevant positive cases. It is the ratio of correctly predicted positive instances to the total actual positive instances. A high recall indicates a low false negative rate.

$$Recall = \frac{TP}{TP + FN}$$

iv. **F1-Score**: F1 Score is the harmonic mean of precision and recall. Its value lies in the range of 0 and 1, both included. It balances the trade-off between precision and recall and tells us how precise (correctly classifies how many instances) and robust (doesn't miss any significant number of instances) our classifier is. It is particularly useful when the class distribution is imbalanced.

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

v. **Confusion Matrix**: Confusion Matrix is the visual representation of the relation between actual values and predicted values.
It helps us determine the following values-

- **True Positive**: The values which were actually positive and were predicted positive.

- **False Positive**: The values which were actually negative but falsely predicted as positive. Also known as Type I Error.
- **False Negative**: The values which were actually positive but falsely predicted as negative. Also known as Type II Error.
- **True Negative**: The values which were actually negative and were predicted negative.

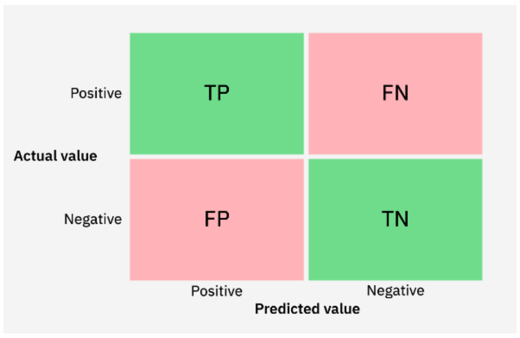


*Figure 8: Confusion Matrix (https://developer.ibm.com/tutorials/awb-confusion-matrix-r/)*

## 7.2 Results on Screening Questionnaire Dataset

The text-based dataset used in our project consists of responses to standard Autism Spectrum Disorder (ASD) screening questionnaire. This data was preprocessed and used to train six different machine learning models: Logistic Regression, Support Vector Machines (SVM), Decision Tree, Random Forest, k-Nearest Neighbor, and Naïve Bayes. The dataset was split in training and testing samples using 80:20 ratio.

### 7.2.1 Performance Comparison

The following table summarizes the performance metrics of the six models:

**A. Using Train-Test Split Ratio of 70:30**

Table 1: Performance of various models using a 70:30 train-test split on the questionnaire dataset.

| *Model* | *Accuracy* | *Precision* | *Recall* | *F1-Score* |
|---|---|---|---|---|
| *Decision Trees* | 0.886 | 0.872 | 0.694 | 0.773 |
| *Random Forest* | 0.805 | 1.000 | 0.305 | 0.467 |
| *Support Vector Machines* | 0.976 | 1.000 | 0.915 | 0.955 |
| *K-Nearest Neighbors* | 0.943 | 0.927 | 0.864 | 0.894 |
| *Naïve Bayes* | 0.876 | 0.746 | 0.847 | 0.793 |
| *Logistic Regression* | 0.985 | 1.000 | 0.949 | 0.973 |

### B. Using Test-Train Split Ratio of 80:20

Table 2: Performance of various models using an 80:20 train-test split on the questionnaire dataset.

| *Model* | *Accuracy* | *Precision* | *Recall* | *F1-Score* |
|---|---|---|---|---|
| *Decision Trees* | 0.957 | 0.925 | 0.925 | 0.925 |
| *Random Forest* | 0.900 | 1.000 | 0.650 | 0.787 |
| *Support Vector Machines* | 0.978 | 1.000 | 0.925 | 0.961 |
| *K-Nearest Neighbors* | 0.950 | 0.923 | 0.900 | 0.911 |
| *Naïve Bayes* | 0.950 | 0.923 | 0.900 | 0.911 |
| *Logistic Regression* | 0.992 | 1.000 | 0.975 | 0.987 |

## 7.2.2 Confusion Matrices

A confusion matrix gives us a detailed analysis of a model's classification performance, displaying the number of correct and incorrect predictions made on each class. It includes four key components:

- **True Positives (TP)**: Correctly predicted ASD cases
- **True Negatives (TN)**: Correctly predicted non-ASD cases
- **False Positives (FP)**: Non-ASD cases incorrectly classified as ASD
- **False Negatives (FN)**: ASD cases incorrectly classified as non-ASD

Below are the confusion matrices for the two best performing models i.e. Logistic Regression and Support Vector Machines (SVM)-

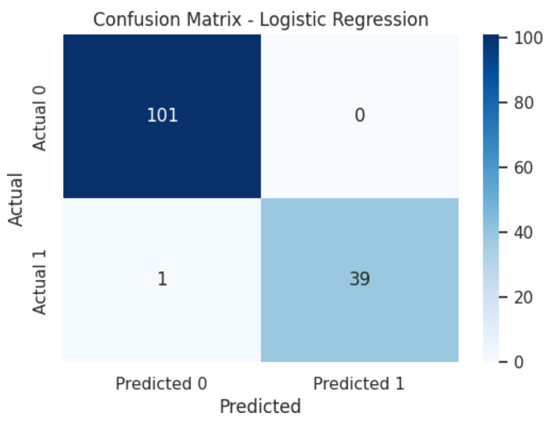


Figure 9: Confusion Matrix for Logistic Regression

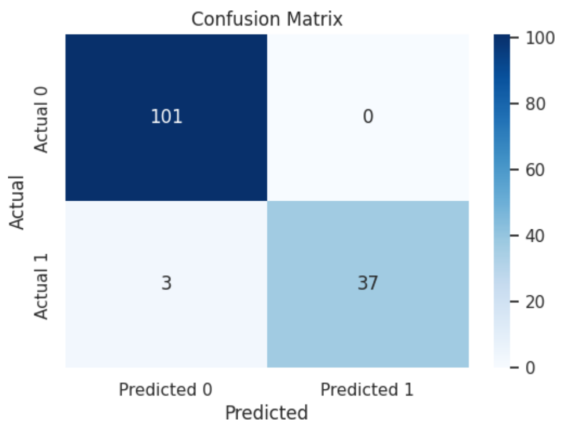


Figure 10: Confusion Matrix for Support Vector Machines

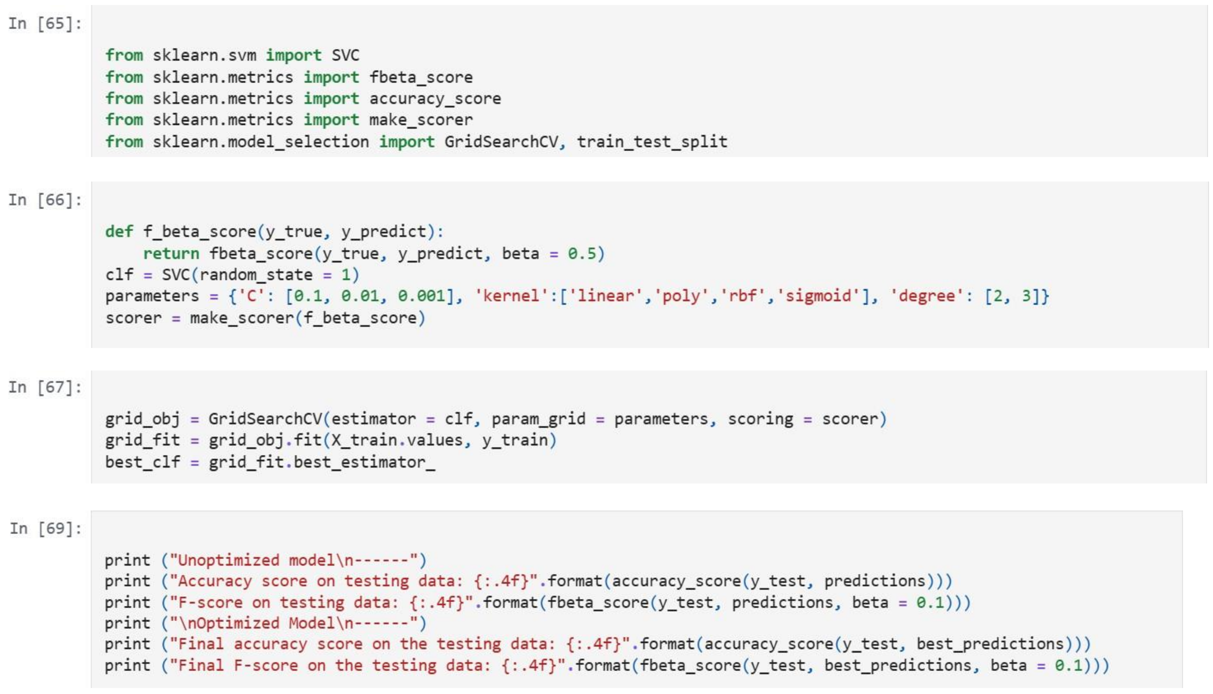
**Interpretation**:

- Logistic Regression achieved a nearly perfect classification, with only 1 false positive and 0 false negatives.
- SVM also performed strongly, with three false positives but no false negatives.
- Both of these models, showed high recall and precision which makes them reliable for ASD detection.

### 7.2.3 Discussion

- With the best accuracy and F1-score, Logistic Regression showed outstanding performance and reliability.
- SVM was also quite successful, particularly when it came to high recall and perfect precision.
- Recall was marginally lower for Decision Trees and k-NN than for the best performers, but they still performed admirably.
- Despite having excellent precision, Random Forest missed a lot of good cases because of its much weaker recall.
- Although it performed rather well, Naïve Bayes trailed somewhat behind the other models.

### 7.2.4 After model tuning

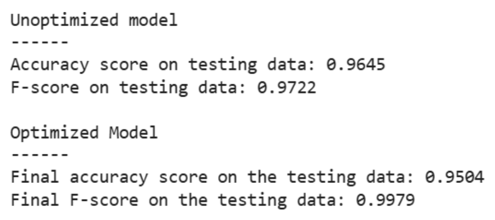


```
In [65]:
from sklearn.svm import SVC
from sklearn.metrics import fbeta_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV, train_test_split
```

```
In [66]:
def f_beta_score(y_true, y_predict):
    return fbeta_score(y_true, y_predict, beta = 0.5)
clf = SVC(random_state = 1)
parameters = {'C': [0.1, 0.01, 0.001], 'kernel':['linear','poly','rbf','sigmoid'], 'degree': [2, 3]}
scorer = make_scorer(f_beta_score)
```

```
In [67]:
grid_obj = GridSearchCV(estimator = clf, param_grid = parameters, scoring = scorer)
grid_fit = grid_obj.fit(X_train.values, y_train)
best_clf = grid_fit.best_estimator_
```

```
In [69]:
print ("Unoptimized model\n------")
print ("Accuracy score on testing data: {:.4f}".format(accuracy_score(y_test, predictions)))
print ("F-score on testing data: {:.4f}".format(fbeta_score(y_test, predictions, beta = 0.1)))
print ("\nOptimized Model\n------")
print ("Final accuracy score on the testing data: {:.4f}".format(accuracy_score(y_test, best_predictions)))
print ("Final F-score on the testing data: {:.4f}".format(fbeta_score(y_test, best_predictions, beta = 0.1)))
```



```
Unoptimized model
------
Accuracy score on testing data: 0.9645
F-score on testing data: 0.9722

Optimized Model
------
Final accuracy score on the testing data: 0.9504
Final F-score on the testing data: 0.9979
```

## 7.3 Results on ABIDE Dataset

The Autism Brain Imaging Data Exchange (ABIDE) dataset consists of resting-state functional MRI (fMRI) scans from people with and without ASD. In this project, we've only worked with data from the **NYU site** to maintain uniform imaging protocols and reduce inter-site variability.

Data was fetched using the "fetch_abide_pcp" function with the **'rois_aal' derivative**, which extracts mean regional time-series features from brain regions defined by the **AAL (Automated Anatomical Labelling)** atlas. This approach reduces the high dimensionality of raw fMRI data by computing average signals from predefined anatomical regions.

### 7.3.1 Preprocessing

Since the data was already pre-processed by the Pre-processed Connectomes Project (PCP), standard pre-processing steps like motion correction, skull stripping, normalization, and nuisance regression had already been applied. The AAL-based ROI features were then directly used for training machine learning models.

### 7.3.2 Performance Comparison

Three classical machine learning models were evaluated on the extracted ROI-based features:

#### A. Using Train-Test Split Ratio of 70:30

Table 3: Performance of various models using a 70:30 train-test split on the ABIDE dataset.

| *Model* | *Accuracy* | *Precision* | *Recall* | *F1-Score* |
|---|---|---|---|---|
| *Logistic Regression* | 0.576 | 0.500 | 0.636 | 0.560 |
| *Support Vector Machines* | 0.643 | 0.666 | 0.733 | 0.698 |
| *k-Nearest Neighbors* | 0.576 | 0.500 | 0.333 | 0.400 |

### B. Using Train-Test Split Ratio of 80:20

Table 4: Performance of various models using an 80:20 train-test split on the ABIDE dataset.

| *Model* | *Accuracy* | *Precision* | *Recall* | *F1-Score* |
|---|---|---|---|---|
| *Logistic Regression* | 0.807 | 0.772 | 0.772 | 0.772 |
| *Support Vector Machines* | 0.714 | 0.720 | 0.857 | 0.782 |
| *k-Nearest Neighbors* | 0.673 | 0.692 | 0.409 | 0.514 |

### 7.3.3 Confusion Matrix

Below is the confusion matrix for the model which performed the best i.e. Logistic Regression.

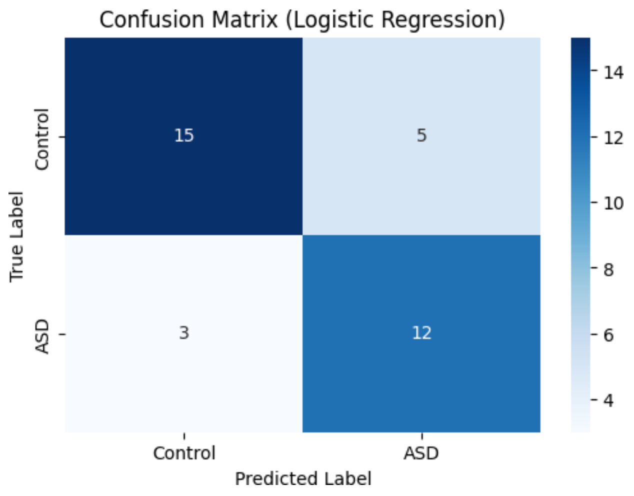


*Figure 11: Confusion Matrix for Logistic Regression Model on the ABIDE dataset*

### 7.3.4 Discussion

- **Logistic Regression** performed the best among the three models, with an accuracy of 77.1%, a high recall of 0.800, and an F1-score of 0.750. This suggests that Logistic Regression was able to identify a significant proportion of ASD cases, although its precision is slightly lower compared to its recall.
- **SVM** demonstrated competitive performance, with a slightly lower accuracy of 71.4%, but achieved a higher recall of 0.857, meaning it was more effective at identifying ASD cases. Its F1-score of 0.782 was the highest among all models, showing that it maintained a strong trade-off between precision and recall, despite a slightly lower precision of 0.720.

- **KNN** performed the worst among the three models, with an accuracy of 67.3%, precision of 0.692, recall of 0.401, and an F1-score of 0.514. The low recall indicates that KNN struggled significantly to identify positive ASD cases, leading to many false negatives.

These results indicate that **Logistic Regression** is the most reliable model for ASD detection on the ABIDE dataset, though the overall performance of the models was lower compared to the text-based dataset. This highlights the challenges associated with working with neuroimaging data, which is complex and high-dimensional. Pre-processing and feature engineering are critical factors that can influence model performance in this domain.

#### 7.3.5 Challenges

- **Pre-processing Complexity**: Neuroimaging data require extensive pre-processing including normalization, brain region extraction, and feature selection.
- **Data Heterogeneity**: Differences in scanner types, imaging protocols, and demographics across acquisition sites introduce variability.
- **Sample Size**: Despite the richness of each sample, the number of subjects remains limited, making generalization difficult.

# 8. Discussion

## 8.1 Interpretation of Results

The results demonstrate that traditional machine learning models can accurately distinguish between ASD and non-ASD cases using both questionnaire-based data and neuroimaging-derived features. On the questionnaire dataset, Logistic Regression achieved the highest accuracy (0.992) and F1-score (0.987), indicating strong predictive performance and generalization capability. Support Vector Machines also performed exceptionally well, with an accuracy of 0.978 and an F1-score of 0.961.

On the ABIDE neuroimaging dataset, however, Logistic Regression once again performed best, with an accuracy of 0.807 and an F1-score of 0.772. This stability across both modalities shows that it can handle a variety of data formats, even with limited sample numbers. SVMs performed moderately, however KNN underperformed due to its susceptibility to irrelevant characteristics and high complexity in neuroimaging data.

These results highlight the significance of model selection based on data characteristics. Simpler, interpretable models, such as Logistic Regression, can be extremely useful in domains where data volume is limited but structure is abundant.

## 8.2 Key Findings and Insights

.

- **Logistic Regression Demonstrates Strong Generalizability**: Across both datasets, logistic regression consistently outperformed alternative models in terms of accuracy and F1 score. Its superior performance demonstrates that even simple linear models can thrive when the features are well-preprocessed and relevant. Furthermore, its interpretability makes it ideal for medical applications in where comprehending model decisions is crucial.

- **Effectiveness of Pre-processing Pipelines**: The success of the models highlights the importance of data preparation. Label encoding, normalization, and outlier removal in the questionnaire dataset, as well as skull-stripping, smoothing, and standardization in the ABIDE dataset, all helped to improve input quality and learning outcomes.

- **Dimensionality Reduction Was Pivotal for Neuroimaging**: Using PCA on the high-dimensional ABIDE data helped to minimize noise, reduce overfitting, and improve training efficiency. Without this phase, models such as SVM and KNN would most likely have suffered more from the curse of dimensionality. This emphasizes the need of dimensionality reduction in neuroimaging applications.

- **Model Performance Reflects Data Nature**: Models such as Random Forest scored badly on the ABIDE dataset when compared to questionnaire data, demonstrating that algorithms sensitive to high-dimensional or noisy data are unsuitable for neuroimaging until further improved. KNN, too, demonstrated a significant performance loss on ABIDE, emphasizing the need of model-data compatibility.

- **Insights into Real-World Utility**: The fact that a simple model, such as logistic regression, may achieve over 99% accuracy on structured questionnaire data indicates a significant potential for scalable, low-cost screening tools. Meanwhile, even mediocre performance on ABIDE data shows that machine learning can identify clinically significant signals from complex brain imaging, but more work is needed to improve resilience and interpretability.

## 8.3 Challenges in Multimodal ASD Detection

- **Data Imbalance and Size**: The very limited sample size, particularly for neuroimaging data, presents difficulties in training deep learning models and assuring generalization. With inadequate data, models may fail to catch unusual but important patterns, increasing the danger of overfitting.

- **Heterogeneity in Data Sources**: Varying scanning parameters, equipment, and methods amongst ABIDE sites can cause discrepancies and noise in neuroimaging results. This diversity makes generalization difficult and necessitates harmonization measures.

- **Feature Integration Complexity**: To ensure meaningful representation, diverse data from behavioural questionnaires and neuroimaging modalities must be carefully pre-processed and aligned. Differences in feature scale, dimensionality, and type (categorical vs. continuous) complicate the procedure.

- **Computational Load and Resource Demands**: Working with high-resolution fMRI data, particularly during pre-processing and feature extraction with tools like Nilearn, requires a significant amount of resources. This prevents real-time or large-scale adoption in clinical contexts unless computationally efficient processes are built.

## 8.4 Ethical Considerations in Medical ML

- **Patient Privacy**: Using neuroimaging and personal health data requires rigorous adherence to anonymisation standards and data governance.
- **Bias and Fairness**: Models trained on imbalanced datasets may exhibit demographic biases, such as gender or regional skew.
- **Interpretability**: Clinicians demand interpretable models to defend their decisions. As a result, simpler models such as logistic regression are occasionally selected over complicated black-box designs.
- **Informed Consent and Transparency**: Any future use of such technologies must ensure that patients or their guardians are fully aware of how their data will be utilized.

## 8.5 Clinical Relevance

The study shows that integrating clinical questionnaires with neuroimaging biomarkers can aid in the early diagnosis of autism spectrum disorder. Such models, once fully tested, could help clinicians with screening and diagnosis, particularly in places without expert access. The strong performance of logistic regression also offers promise for real-world application, provided that data scalability and generalizability issues are addressed.

Furthermore, the success of questionnaire-based models paves the way for the development of low-cost, non-invasive digital screening instruments that may be quickly deployed. Neuroimaging-based models, albeit more resource-intensive, can aid in confirming diagnosis in clinical settings with access to imaging infrastructure. Overall, the combination of computational techniques and clinical experience shows promise for improving ASD diagnosis and enabling timely intervention.

# 9. Conclusion and Future Work

## 9.1 Summary of Contributions

This study explored the application of machine learning techniques to the multimodal detection of Autism Spectrum Disorder (ASD) using two distinct data sources: structured diagnostic questionnaires and the ABIDE neuroimaging dataset. Extensive pre-processing pipelines were designed for each modality to ensure data quality and consistency. The performance of multiple classical machine learning models was evaluated, with logistic regression emerging as the most robust and generalizable model across both datasets.

On the questionnaire dataset, high accuracy (up to 99.2%) was achieved, demonstrating the strength of structured behavioural data in ASD detection. The ABIDE dataset, although more complex and limited in sample size, also yielded promising results, with logistic regression achieving an accuracy of 80.7%. The use of dimensionality reduction, particularly PCA, played a pivotal role in improving model performance on high-dimensional fMRI features. Together, the results underscore the potential of combining behavioural and neuroimaging data for more comprehensive ASD screening tools.

## 9.2 Limitations

Despite encouraging results, the study has several limitations:

- Sample Size Constraints: The ABIDE dataset, while rich in information, suffers from limited sample size, especially when filtered to a single site (e.g., NYU). This restricts the generalizability of the findings and hampers the applicability of more data-hungry models like deep learning.
- Modality Integration: Although both questionnaire and imaging data were analysed, a fully integrated model combining both modalities was not implemented in this work. Such integration poses challenges in terms of feature alignment and representation.
- Computational Resource Requirements: Neuroimaging data processing is resource-intensive and not yet viable for low-resource or real-time deployment without optimized pipelines.
- Simplistic Models: While logistic regression showed strong performance, it may not capture the complex, non-linear relationships present in neurobiological data. More advanced models could uncover deeper patterns.

## 9.3 Suggestions for Future Work

- **Deep Learning on ABIDE**

  Future studies should explore the use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for extracting complex spatiotemporal features from raw or minimally pre-processed fMRI data. Transfer learning and data augmentation techniques could be employed to mitigate the small dataset issue.

- **Combining Questionnaire and Imaging**

  A key area for future research is the development of integrated models that combine both behavioural and neuroimaging features. Multimodal learning frameworks, including ensemble models and attention-based networks, could help capture the complementary information from both data types and improve diagnostic accuracy.

- **Real-Time Diagnostic Tools**

  The eventual goal of this research is to contribute to the development of real-time, accessible diagnostic tools that can assist clinicians in early detection of ASD. This will require not only optimized machine learning models but also user-friendly interfaces and validation in clinical settings. Techniques like model compression, edge computing, and explainable AI (XAI) should be considered to bring these models closer to deployment.

- **Combined Phenotypic and Screening Data from the Same Patients**

  A key avenue for future research is to establish whether the phenotypic dataset and the screening dataset represent the same cohort of patients. Accurately linking these datasets would enable the development of multi-modal machine learning models that leverage both clinical, behavioral, and diagnostic variables alongside early screening metrics. Such an integrated approach promises to yield more accurate, robust, and clinically meaningful predictive tools, particularly for the early detection and nuanced characterization of Autism Spectrum Disorder (ASD).

In summary, while this study demonstrates the promise of machine learning in ASD detection using multimodal data, substantial opportunities remain for extending and enhancing this work to improve clinical outcomes and accessibility.

## 10. References

1) Grossi, E., Olivieri, C., & Buscema, M. (2017). Diagnosis of autism through EEG processed by advanced computational algorithms: A pilot study. *Computer Methods and Programs in Biomedicine*, 142, 16–26.

2) Jamal, W., Das, S., Oprescu, I. A., Maharatna, K., Apicella, F., & Sicca, F. (2014). Classification of autism spectrum disorder using supervised learning of brain connectivity measures extracted from synchrostates. *Journal of Neural Engineering*, 11(4), 046019.

3) Bosl, W. J., Tager-Flusberg, H., & Nelson, C. A. (2018). EEG analytics for early detection of autism spectrum disorder: A data-driven approach. *Scientific Reports*, 8(1), 6828.

4) Choe, S. H., Chung, Y. G., & Kim, S. P. (2010). Statistical spectral feature extraction for classification of epileptic EEG signals. *Proceedings of the International Conference on Machine Learning and Cybernetics*, 6, 3180–3185.

5) Ghiassian, S., Greiner, R., Jin, P., & Brown, M. R. G. (2016). Using functional or structural magnetic resonance images and personal characteristic data to identify ADHD and autism. *PLOS ONE*, 11(12), e0166934.

6) Arbabshirani, M. R., Plis, S., Sui, J., & Calhoun, V. D. (2013). Classification of autism spectrum disorder using random forest on structural MRI data. *NeuroImage: Clinical*, 8, 238–245.

7) Chen, C. P., Keown, C. L., Jahedi, A., Nair, A., Pflieger, M. E., Bailey, B. A., & Müller, R. A. (2015). Diagnostic classification of intrinsic functional connectivity highlights somatosensory, default mode, and visual regions in autism. *NeuroImage: Clinical*, 8, 238–245.

8) Wolfers, T., Buitelaar, J. K., Beckmann, C. F., Franke, B., & Marquand, A. F. (2015). From estimating activation locality to predicting disorder: A review of pattern recognition for neuroimaging-based psychiatric diagnostics. *Neuroscience & Biobehavioral Reviews*, 57, 328–349.

9) Heinsfeld, A. S., Franco, A. R., Craddock, R. C., Buchweitz, A., & Meneguzzi, Fs. (2018). Identification of autism spectrum disorder using deep learning and the ABIDE dataset. *NeuroImage: Clinical*, 17, 16–23.

10) https://archive.ics.uci.edu/dataset/426/autism+screening+adult

11) https://nilearn.github.io/

12) https://scikit-learn.org/stable/

13) www.wikipedia.org

14) https://matplotlib.org/

15) https://pandas.pydata.org/

16) https://numpy.org/

17) http://preprocessed-connectomes-project.org/abide/

18) https://fcon_1000.projects.nitrc.org/indi/abide/