1. Introduction

Autism or Autism Spectrum Disorder (ASD), is a neurological disease either in children or adult (Nurul Amirah Mashudi et al., 2021), which is also a wide range of condition characterized by challenges with speech, social skills, repetitive behavior and non-verbal communication(*Autism Spectrum Disorder (ASD) | Autism Speaks*, 2024). The signs can be detected as early as 1 year old *(Pierce et al., 2011)*, includes no response to name, poor skills in imitation, problems with eye contact and joint attention (*When Do Children Usually Show Symptoms of Autism?*, 2017).

Meanwhile, adult autism is way more complex than toddler as the symptoms can be present differently for adults of different genders. The common signs adult ASD includes difficulties in making friends, anxiety in social situations, problems in understanding what others are thinking or feeling, struggle in describing own feelings (NHS Choices, 2024). Women are often diagnosed later in life due to the development of compensatory strategies to mask their challenges, such as strong imitation skills to "camouflage" themselves into understanding social norms, therefore make their ASD traits undetectable in everyday interactions (*How Is Autism Different in Women?*, 2023).

In the data set, the score of each question in AQ-10 is recorded along with details of the person such as age, sex, and ethnicity. The Autism Spectrum Quotient (AQ) is a questionnaire designed for adults (16 years old and above) who do not have learning disability, AQ-10 is used as a screening tool to determine whether a person should be referred for an autism assessment *(Engelbrecht, 2020)*.

The models used for predicting autism/ASD involved Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbour (KNN) and VotingClassifier in Ensemble Learning. The result from each model and their evaluation would be recorded as a comparison to determine which is the best model.

2. Problem Formulation

2.1 Predict autism based on the AQ-10 answer

The AQ-10 score of 6 and above indicates autism or number of autistic traits, while lower than 6 means the person might not be autistic. Hence, the accuracy of the AQ-10 test is the subject of study.

2.2 Predict autism from all the features given

Further details such as age, sex, gender, ethnicity, and presence of jaundice are added to enhance the result of prediction. The additional variables would potentially increase the accuracy of prediction on top of the AQ-10 answers, as there is a chance to discover the hidden patterns in the matrix of data points.

3. Problem and Data Understanding

3.1 Problem Understanding

3.1.1 Ideal solution for the problem

The model is expected to be able to predict ASD or autism up to accuracy of 80%.

3.1.2 Characteristics for the ideal solution

Use a classifier to classify whether the case is autistic or not.

3.1.3 What type of data mining modelling must perform?

Classification models which are K-Nearest Neighbour (KNN), ensemble learning methods, RF and LR are used to predict the result in target variable “Class/ASD”, on whether autism is “Yes” or “No”.

3.1.4 Range of estimates

The output in target variable is “Yes” for positive in autism, and “No” for not autistic.

3.2 Data Understanding

3.2.1 Data collection

Data descriptions (Umar, 2024):

Features	**Type**	**Description**	**Expected value**
A1_Score	Binary	Answer code of question in AQ-10	0 or 1
A2_Score	Binary	Answer code of question in AQ-10	0 or 1
A3_Score	Binary	Answer code of question in AQ-10	0 or 1
A4_Score	Binary	Answer code of question in AQ-10	0 or 1
A5_Score	Binary	Answer code of question in AQ-10	0 or 1
A6_Score	Binary	Answer code of question in AQ-10	0 or 1
A7_Score	Binary	Answer code of question in AQ-10	0 or 1
A8_Score	Binary	Answer code of question in AQ-10	0 or 1
A9_Score	Binary	Answer code of question in AQ-10	0 or 1
A10_Score	Binary	Answer code of question in AQ-10	0 or 1
age	Number	Age of case	Minimum of 17 years old
gender	String	Male or Female	f or m

ethnicity	String	Ethnicity of participants	White-European, Latino, Black, Asian, Middle Eastern, Pasifika, South Asian, Hispanic, Turkish, Others
jaundice	Boolean	If the case was born with jaundice	Yes or no
austim	Boolean	If immediate family member has been diagnosed with autism	Yes or no
contry_of_res	String	Where the participant resides	United States, Bahamas, etc
used_app_before	Boolean	ASDTest app or not	Yes or no
result	Integer	Score for AQ1-10 screening test	Value ranged from 1 to 10
age_desc	String	Age category of the case	18 and more
relation	String	Relation of patient who completed the test	Self, Parent, Health Care Professional, Relative, Others
Class/ASD	Boolean	Target variable, whether the case has autism or not	Yes or No

3.2.2 Data sample

A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jundice	austim	contry_of_r	used_app_b	result	age_desc	relation	Class/ASD
1	1	1	1	0	0	1	1	0	0	26	f	White-Euro	no	no	United State	no	6	18 and mor	Self	NO
1	1	0	1	0	0	0	1	0	1	24	m	Latino	no	yes	Brazil	no	5	18 and mor	Self	NO
1	1	0	1	1	0	1	1	1	1	27	m	Latino	yes	yes	Spain	no	8	18 and mor	Parent	YES
1	1	0	1	0	0	1	1	0	1	35	f	White-Euro	no	yes	United State	no	6	18 and mor	Self	NO
1	0	0	0	0	0	0	1	0	0	40	f	?	no	no	Egypt	no	2	18 and mor	?	NO
1	1	1	1	1	0	1	1	1	1	36	m	Others	yes	no	United State	no	9	18 and mor	Self	YES
0	1	0	0	0	0	0	1	0	0	17	f	Black	no	no	United State	no	2	18 and mor	Self	NO
1	1	1	1	0	0	0	0	1	0	64	m	White-Euro	no	no	New Zealan	no	5	18 and mor	Parent	NO
1	1	0	0	1	0	0	1	1	1	29	m	White-Euro	no	no	United State	no	6	18 and mor	Self	NO
1	1	1	1	0	1	1	1	1	0	17	m	Asian	yes	yes	Bahamas	no	8	18 and mor	Health care	YES
1	1	1	1	1	1	1	1	1	1	33	m	White-Euro	no	no	United State	no	10	18 and mor	Relative	YES
0	1	0	1	1	1	1	0	0	1	18	f	Middle East	no	no	Burundi	no	6	18 and mor	Parent	NO
0	1	1	1	1	1	0	0	1	0	17	f	?	no	no	Bahamas	no	6	18 and mor	?	NO
1	0	0	0	0	0	1	1	0	1	17	m	?	no	no	Austria	no	4	18 and mor	?	NO
1	0	0	0	0	0	1	1	0	1	17	f	?	no	no	Argentina	no	4	18 and mor	?	NO
1	1	0	1	1	0	0	1	0	1	18	m	Middle East	no	yes	New Zealan	no	6	18 and mor	Parent	NO
1	0	0	0	0	0	1	1	1	1	31	m	Middle East	no	no	Jordan	no	5	18 and mor	Self	NO
0	0	0	0	0	0	0	1	0	1	30	m	White-Euro	no	no	Ireland	no	2	18 and mor	Self	NO
0	0	1	0	1	1	0	0	0	0	35	f	Middle East	no	yes	United Arab	no	3	18 and mor	Self	NO

3.2.3 What is the data size

The data set “autism_screening.csv” consists of 21 columns and 704 records.

3.2.4 How the data is stored

The data set is stored in Comma separated values (CSV) and Attribute-Relation File Format (ARFF) format.

3.2.5 Data source

The Autism Screening on Adults (Larxel, 2017) dataset is obtained from Kaggle. Similar dataset with title Autism Screening Adult (UCI Machine Learning Repository, 2017) is found on another source. Features of anonymous participants details are from ASDTests screening app (Fadi, 2018), while the 10 AQ answer are based on AQ-10 adult (Allison et al., 2011).

4. Data Pre-processing **Methods**

The issues found in the data set and steps taken for preprocessing are shown:

Blank space in columns and values

a. Column: Each of the columns has a variety of blank spaces behind the last character. Therefore, *replace()* method is used for removing those spaces, by replacing each space with an empty string.

```
# check column names
df.columns
✓ 0.0s
```

```
Index(['A1_Score ', 'A2_Score ', 'A3_Score ', 'A4_Score ', 'A5_Score ',
       'A6_Score ', 'A7_Score ', 'A8_Score ', 'A9_Score ', 'A10_Score ',
       'age   ', 'gender ', 'ethnicity      ', 'jundice ', 'austim ',
       'contry_of_res        ', 'used_app_before ', 'result ', 'age_desc    ',
       'relation             ', 'Class/ASD'],
      dtype='object')
```

```
# remove column space
df.columns = df.columns.str.replace(' ', '')
df.columns
✓ 0.0s
```

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_app_before',
       'result', 'age_desc', 'relation', 'Class/ASD'],
      dtype='object')
```

b. Values: There are also spaces behind values under the columns, such as "ethnicity". Effect of before and after the cleaning is shown.

```
df['ethnicity'].unique()
✓ 0.0s
```

```
array(['White-European ', 'Latino         ', '?              ',
       'Others         ', 'Black          ', 'Asian          ',
       'Middle Eastern ', 'Pasifika       ', 'South Asian    ',
       'Hispanic       ', 'Turkish        ', 'others         '],
      dtype=object)
```

```
# trim the spaces
df = df.map(lambda x: x.strip() if isinstance(x, str) else x)
df['ethnicity'].unique()
✓ 0.0s
```

```
array(['White-European', 'Latino', '?', 'Others', 'Black', 'Asian',
       'Middle Eastern', 'Pasifika', 'South Asian', 'Hispanic', 'Turkish',
       'others'], dtype=object)
```

Typo on columns' name

Column name: *rename()* method is used for renaming the column names with typo, which are "austim", "contry_of_res" and "jundice". The names are corrected after running commands.

```
# column typo
df.rename(columns={
    "austim": "autism",
    "contry_of_res":"country_of_res",
    "jundice":"jaundice"}, inplace=True)

df.columns
```

✓ 0.0s MagicPython

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jaundice', 'autism', 'country_of_res', 'used_app_before',
       'sum_score', 'age_desc', 'relation', 'Class/ASD'],
      dtype='object')
```

Rename column to avoid confusion:

"result" column refers to sum of AQ score, and "autism" refers to family members diagnosed with autism, hence the columns shall be named accordingly.

```
# change column name
df.rename(columns={
    "autism": "fam_with_autism",
    "result": 'sum_score'}, inplace=True)

df.columns
```

✓ 0.0s MagicPython

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jaundice', 'fam_with_autism', 'country_of_res',
       'used_app_before', 'sum_score', 'age_desc', 'relation', 'Class/ASD'],
      dtype='object')
```

Mismatch values

Through running the *unique()* method for each column, it is found that "ethnicity" and "relation" columns have odd value of "?".

```python
# check if there are any out of place values
for row in df.columns:
    print(row, df[row].unique())
```
✓ 0.0s MagicPython

```
A1_Score [1 0]
A2_Score [1 0]
A3_Score [1 0]
A4_Score [1 0]
A5_Score [0 1]
A6_Score [0 1]
A7_Score [1 0]
A8_Score [1 0]
A9_Score [0 1]
A10_Score [0 1]
age ['26.0' '24.0' '27.0' '35.0' '40.0' '36.0' '17.0' '64.0' '29.0' '33.0'
 '18.0' '31.0' '30.0' '34.0' '38.0' '42.0' '43.0' '48.0' '37.0' '55.0'
 '50.0' '53.0' '20.0' '28.0' '21.0' '383.0' '47.0' '32.0' '44.0' '' '19.0'
 '58.0' '45.0' '22.0' '39.0' '25.0' '23.0' '54.0' '60.0' '41.0' '46.0'
 '56.0' '61.0' '59.0' '52.0' '49.0' '51.0']
gender ['f' 'm']
ethnicity ['White-European' 'Latino' '?' 'Others' 'Black' 'Asian' 'Middle Eastern'
 'Pasifika' 'South Asian' 'Hispanic' 'Turkish' 'others']
jaundice ['no' 'yes']
fam_with_autism ['no' 'yes']
country_of_res ['United States' 'Brazil' 'Spain' 'Egypt' 'New Zealand' 'Bahamas'
 'Burundi' 'Austria' 'Argentina' 'Jordan' 'Ireland' 'United Arab Emirates'
 'Afghanistan' 'Lebanon' 'United Kingdom' 'South Africa' 'Italy'
 'Pakistan' 'Bangladesh' 'Chile' 'France' 'China' 'Australia' 'Canada'
 'Saudi Arabia' 'Netherlands' 'Romania' 'Sweden' 'Tonga' 'Oman' 'India'
...
sum_score [ 6.  5.  8.  2.  9. 10.  4.  3.  0.  1.  7.]
age_desc ['18 and more']
relation ['Self' 'Parent' '?' 'Health care professional' 'Relative' 'Others']
Class/ASD ['NO' 'YES']
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

By using *shape* method, 95 records in total have "?". Then, filter by *loc* shows that "ethnicity" and "relation" has overlap in the occurrence of "?", that means if "ethnicity" has "?", "relation" surely would have it too, or vice versa.

```python
print("relation", df[df['relation']=='?'].shape)
print("ethnicity", df[df['ethnicity']=='?'].shape)
```
✓ 0.0s

```
relation (95, 21)
ethnicity (95, 21)
```

```python
df.loc[df['ethnicity'].eq("?") & df['relation'].eq("?")].shape
```
✓ 0.0s

```
(95, 21)
```

Since there are only 95 out of 704 records with this unknown value, it is safe to use *drop()* method to remove these records. After the removal, there are 609 records left.

```python
# remove '?' values
df.drop(df.loc[df['relation']=='?'].index, inplace=True)
df.shape
```
✓ 0.0s

```
(609, 21)
```

Standardize “ethnicity” values

From *unique()* method, the “Others” and “others” differ by a capital letter; “Turkish” is considered as “Middle Eastern”, while “Latino” and “Hispanic” do have some share traits but can be geographically distributed, same goes to “Asian” and “South Asian” whom might have contrasting genetics. Thus, only “others” and “Turkish” are being replaced. Also, a hyphen is removed from “White-European”.

```
df['ethnicity']=df['ethnicity'].replace({
    'others':'Others',
    'Turkish':'Middle Eastern',
    'White-European':'White European'})
df['ethnicity'].unique()
```

✓ 0.0s

```
array(['White European', 'Latino', 'Others', 'Black', 'Asian',
       'Middle Eastern', 'Pasifika', 'South Asian', 'Hispanic'],
      dtype=object)
```

Wrong data types

The values under “age” and “sum_score” column should be integer as there is only 0 after decimal. Data types of these columns are changed to “float64” first, then to “int64”.

```
# After change dtypes
df[['age','sum_score']]=df[['age','sum_score']].astype('float64').astype('int64')
df.dtypes
```

✓ 0.0s MagicPython

```
A1_Score           int64
A2_Score           int64
A3_Score           int64
A4_Score           int64
A5_Score           int64
A6_Score           int64
A7_Score           int64
A8_Score           int64
A9_Score           int64
A10_Score          int64
age                int64
gender            object
ethnicity         object
jaundice          object
fam_with_autism   object
country_of_res    object
used_app_before   object
sum_score          int64
age_desc          object
relation          object
Class/ASD         object
dtype: object
```

```
df[['age','sum_score']].head()
```

✓ 0.0s

	age	sum_score
0	26	6
1	24	5
2	27	8
3	35	6
5	36	9

The “age” and “sum_score” do not have decimal place now.

Remove unused column

Under "age_desc" column, "18 and more" is the only value, therefore this column does not have any variety. So, this column can be removed by using *drop()* method

```
# check unique value under age_desc
df["age_desc"].unique()
```

✓ 0.0s MagicPython

```
array(['18 and more'], dtype=object)
```

```
# remove unused column: age_desc
df.drop("age_desc",axis=1,inplace=True)
df.columns
```

✓ 0.0s MagicPython

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jaundice', 'fam_with_autism', 'country_of_res',
       'used_app_before', 'sum_score', 'relation', 'Class/ASD'],
      dtype='object')
```

Create new data sets

New data frame containing AQ-10 results and target variable is created, and saved as a new data set with CSV format, under name "score.csv"

Cleaned version of data frame containing all features except "age_desc" also saved as "cleaned_autism_screening.csv"

```
# Create DF for AQ-10 only
score_df = df.iloc[:, 0:10]
score_df[['sum', 'target']] = df[['sum_score', 'Class/ASD']]
score_df.head()
```

✓ 0.0s MagicPython

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	sum	target
0	1	1	1	1	0	0	1	1	0	0	6	NO
1	1	1	0	1	0	0	0	1	0	1	5	NO
2	1	1	0	1	1	0	1	1	1	1	8	YES
3	1	1	0	1	0	0	1	1	0	1	6	NO
5	1	1	1	1	1	0	1	1	1	1	9	YES

```
# Save as CSV
score_df.to_csv("score.csv", index = False)
df.to_csv("cleaned_autism_screening.csv", index = False)
```

✓ 0.0s MagicPython

5. Data Modelling & Evaluation

Models

The prediction of autism / ASD from data set is a classification problem, therefore, the techniques used are:

a. Ensemble learning - VotingClassifier with Logistic regression, Decision Tree, and Support Vector as sub models
   - Advantages: Has more resistance towards incorrect or impacts from individual model's errors through taking various viewpoints into account. Also, accuracy can be improved via reduced variance and bias to enhance the overall performance by leveraging the collective knowledge of multiple models, therefore promoting higher accuracy in prediction. (*SoulPage IT Solutions*, 2023)
   - Disadvantages: There is a risk of information loss as the confidence of individual models is being ignored. (Syed Wahaj, 2023)

b. K-Nearest Neighbours (KNN)
   - Advantages: Due to the simplicity of algorithm, it has high accuracy with less hyper-parameter, which are k value and distance metric. In terms of adaptability, this algorithm could adjust itself for new data as training data already stored into memory.
   - Disadvantages: KNN has low scalability as it requires more memory, thus needs longer time for computation. The algorithm is also guilty of performance issues with high-dimensional data input, called peaking phenomenon. (IBM, 2021)
   - The disadvantages of KNN can be ignored as the size of the data set is small (21 columns, 608 records), and there would be no additional features or records to be added for extra training and testing.

c. Random Forest (RF)
   - Advantages: Parallel processing is supported as the trees are created in parallel, speeding up training time since the iterations are independent. The model can handle vast amount data with high dimension and is valuable in understanding the hidden patterns by providing information about the importance of each feature in data.

- Disadvantages: Possess a degree of challenge in making interpretation as many decision trees are involved. Computational complexity is also another factor as it needs lots of memory when comes to large datasets. (AIML.com, 2022)

d. Logistic regression (LR)

- Advantages: Fast and able to handle data sets of large dimensions. Due to its simplicity, the result in the form of coefficients is not complex for interpretation.
- Disadvantages: The sensitivity towards outliers could affect the coefficients and predictions. The model also has risk of over-fitting when it comes to feature rich data set. (AIML.com, 2022)

Evaluation

Regression and classification have different sets of metrics in evaluating the performance of their models. The set of metrics included for classification are:

a. Confusion Matrix:

		Actual	
		Y	N
Predicted	Y	True Positive (TP)	False Positive (FP)
	N	False Negative (FN)	True Negative (TN)

- Accuracy: correct predictions out of total observations.

$$\frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: positive prediction among all positively predicted actual cases.

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

- Recall or sensitivity: true positive rate (Accuracy vs. Precision vs. Recall in Machine Learning: What’s the Difference?, 2024), positive prediction among all actual positive cases.

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

- Specificity: true negative rate, negative prediction among all actual negative cases.

$$\frac{TN}{TN + FP}$$

- Misclassification: is the complement of accuracy, calculated by 1-accuracy. Lower value is preferred

b. Receiver Operator Characteristic (ROC curves): This method works by plotting true positive versus false positive rate. The maximum accuracy is 1.0 or 100%. The models would be plotted on the same ROC curve for comparison. The best performed model would be close to 100% or have the largest area under the curve (AUC) value.

6. Data Modelling and Evaluation phases

Before the modelling process, the features with string values would undergo *one-hot encoding* technique to convert categorical variables into numerical values (GeeksforGeeks, 2019). While the "Yes" and "No" values would be translated into 1 for "Yes", and 0 for "No". Next, each set of data is splitted into 75% for training and 25% for testing. Those training and testing data sets would undergo feature scaling to standardise the range of data in features (Christos Goumopoulos & Stergiopoulos, 2022).

score.csv

Repeated steps for all models: library import, data splitting and feature scaling

```
# import libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')

# split score_df
score_df = pd.read_csv("score.csv")
X = pd.get_dummies(score_df[[
        'A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score',
        'A6_Score','A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'sum']])
Y = score_df['target'].map({'NO':0,'YES':1}) # convert to 0, 1 for AUC calculation

#Splitting dataset into training and testing dataset
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(
    X, Y, stratify=Y, test_size=0.25, random_state=50)

# feature scaling
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

<u>Ensemble Learning - VotingClassifier</u>

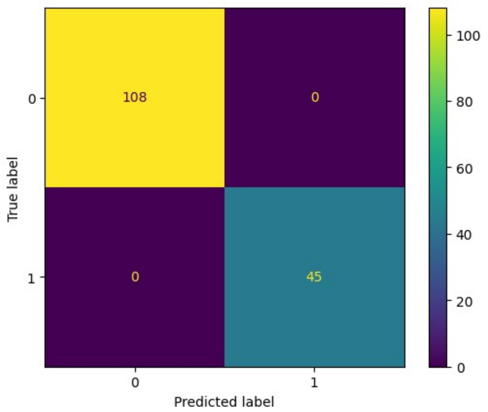
Use LR, decision tree, and support vector classifier as the models for voting.

```
# Ensemble Learning - Voting
# Voting-based Ensemble for Classification using various methods
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier

# create the sub models using LogisticRegression, DecisionTree, and SupportVector
estimators = []
model1 = LogisticRegression(random_state=7)  # Logistic Regression Classifier
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier(random_state=7) # Decision Tree Classifier
estimators.append(('cart', model2))
model3 = SVC(random_state=7) #Support Vector Classifier
estimators.append(('svc', model3))

# create the ensemble model
es_Vote = VotingClassifier(estimators)
es_Vote.fit(X_train, Y_train)
y_predict = es_Vote.predict(X_test)
```

Run Confusion Matrix to get values for accuracy, precision, recall, specificity, and misclassification. Also, use *roc_auc_score()* to find AUC.

```
# Model performance
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay

# Confusion Matrix
cm_en=confusion_matrix(Y_test, y_predict_en, labels=es_Vote.classes_)
en_TP = cm_en[0][0]
en_FP = cm_en[0][1]
en_FN = cm_en[1][0]
en_TN = cm_en[1][1]

print("TP: ", en_TP, '\nFP: ', en_FP, '\nFN: ', en_FN, '\nTN: ', en_TN)
print(es_Vote.__class__.__name__,
      '\nEnsemble accuracy: ', accuracy_score(Y_test, y_predict_en),
      '\nEnsemble precision: ',precision_score(Y_test, y_predict_en), #TP/(TP+FP),
      '\nEnsemble recall: ', recall_score(Y_test, y_predict_en) , #TP/(TP+FN),
      '\nEnsemble specificity: ', en_TN/(en_TN+en_FP),
      '\nEnsemble misclassification: ', 1-accuracy_score(Y_test, y_predict_en)
      )

en_matrix = ConfusionMatrixDisplay(
    confusion_matrix=cm_en,
    display_labels=es_Vote.classes_)
en_matrix.plot()

# ROC - AUC
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

en_auc = roc_auc_score(Y_test, y_predict_en)
print("Ensemble AUC: ", en_auc)
```

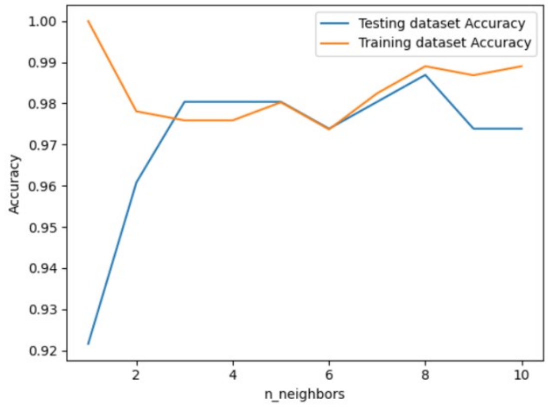
```
TP:  108
FP:  0
FN:  0
TN:  45
VotingClassifier
Ensemble accuracy:  1.0
Ensemble precision:  1.0
Ensemble recall:  1.0
Ensemble specificity:  1.0
Ensemble misclassification:  0.0
Ensemble AUC:  1.0
```



KNN

Set a range of k-Neighbours from 1 to 10 to find out the k-value for highest accuracy in testing data set. From the graph, the overlap accuracy between testing and training set is at k=5 and k=6, while the accuracy for testing is highest at 8.

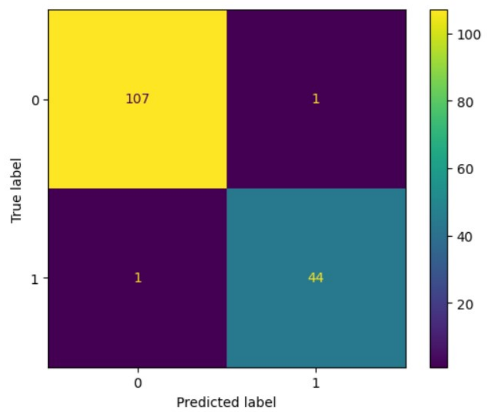
```python
# Trial - find out which k-value is the best
neighbors = np.arange(1, 11)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train)

    # Compute training and test data accuracy
    train_accuracy[i] = knn.score(X_train, Y_train)
    test_accuracy[i] = knn.score(X_test, Y_test)

plt.plot(neighbors,
         test_accuracy,
         label = 'Testing dataset Accuracy')
plt.plot(neighbors,
         train_accuracy,
         label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



Therefore, k=8 is used for getting the values in Confusion matrix, and AUC.

```python
# use k = 8
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, Y_train)

y_predict_knn = knn.predict(X_test)
knn.score(X_test, Y_test)

cm_knn = confusion_matrix(Y_test, y_predict_knn)
knn_TP = cm_knn[0][0]
knn_FP = cm_knn[0][1]
knn_FN = cm_knn[1][0]
knn_TN = cm_knn[1][1]

print(knn.__class__.__name__,
      '\nKNN accuracy: ', round(accuracy_score(Y_test, y_predict_knn), 4),
      '\nKNN precision: ',round(precision_score(Y_test, y_predict_knn),4), # TP/(TP+FP),
      '\nKNN recall: ', round(recall_score(Y_test, y_predict_knn), 4), # TP/(TP+FN),
      '\nKNN specificity: ', round(knn_TN/(knn_TN+knn_FP), 4),
      '\nKNN misclassification: ', round(1-accuracy_score(Y_test, y_predict_knn), 4)
      )

knn_matrix = ConfusionMatrixDisplay(confusion_matrix=cm_knn, display_labels=knn.classes_)
knn_matrix.plot()

# ROC - AUC
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

knn_auc = roc_auc_score(Y_test, y_predict_knn)
print("KNN AUC: ", round(knn_auc, 4))
```

```
KNeighborsClassifier
KNN accuracy:  0.9869
KNN precision:  0.9778
KNN recall:  0.9778
KNN specificity:  0.9778
KNN misclassification:  0.0131
KNN AUC:  0.9843
```

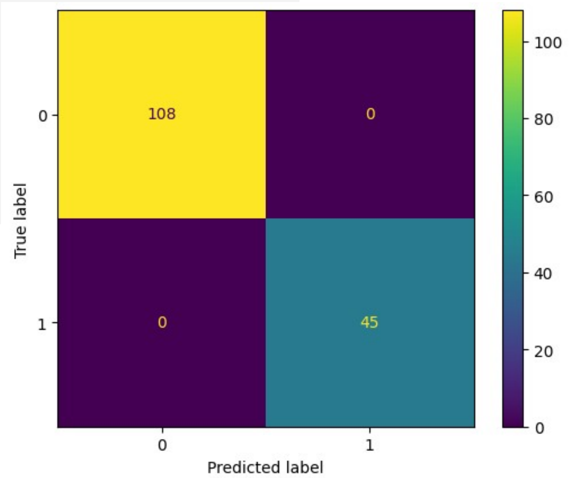


RF

The result of using *RandomForestClassifier()* is the same as *VotingClassifier(),* where the accuracy and AUC have the ideal value of 1.0. Hence, there is no improvement needed to enhance the model's performance.

```python
# Random Forest Classifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, Y_train)

y_predict_rf = rf.predict(X_test)

cm_rf = confusion_matrix(Y_test, y_predict_rf)
rf_TP = cm_rf[0][0]
rf_FP = cm_rf[0][1]
rf_FN = cm_rf[1][0]
rf_TN = cm_rf[1][1]

print("TP: ", rf_TP, '\nFP: ', rf_FP, '\nFN: ', rf_FN, '\nTN: ', rf_TN)
print(rf.__class__.__name__,
      '\nRandom Forest accuracy: ', round(accuracy_score(Y_test, y_predict_rf), 4),
      '\nRandom Forest precision: ',round(precision_score(Y_test, y_predict_rf),4), # TP/(TP+FP),
      '\nRandom Forest recall: ', round(recall_score(Y_test, y_predict_rf), 4), # TP/(TP+FN),
      '\nRandom Forest specificity: ', round(rf_TN/(rf_TN+rf_FP), 4),
      '\nRandom Forest misclassification: ', round(1-accuracy_score(Y_test, y_predict_rf), 4)
      )

rf_matrix = ConfusionMatrixDisplay(
    confusion_matrix=cm_rf,
    display_labels=rf.classes_)
rf_matrix.plot()

# ROC - AUC
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

rf_auc = roc_auc_score(Y_test, y_predict_rf)
print("Random Forest AUC: ", round(rf_auc, 4))
```

```
TP:  108
FP:  0
FN:  0
TN:  45
RandomForestClassifier
Random Forest accuracy:  1.0
Random Forest precision:  1.0
Random Forest recall:  1.0
Random Forest specificity:  1.0
Random Forest misclassification:  0.0
Random Forest AUC:  1.0
```



LR

LogisticRegression() is applied with *solver* = *'liblinear'* for binary classification with small dataset.

```python
# Logistic Regression
import numpy as np
from sklearn.linear_model import LogisticRegression

lg = LogisticRegression(solver='liblinear', random_state=50, max_iter=300)
lg.fit(X_train, Y_train)

# obtain the model intercept and coefficient of each input attribute
print(f"intercept: {np.round(lg.intercept_,4)}")
fieldList = np.array(list(X)).reshape(-1,1)
coeffs = np.reshape(np.round(lg.coef_,2),(-1,1))
coeffs=np.concatenate((fieldList,coeffs),axis=1)
print(pd.DataFrame(coeffs,columns=['Attribute','Coefficient']))

# fit the model to compare predicted and actual target output values
y_predict_lg = lg.predict(X_test)
```

LR result:

$$\begin{aligned} \text{target} = {} & 0.98 * \text{A1\_Score} + 1.05 * \text{A2\_Score} + 0.81 * \text{A3\_Score} + 1.03 \\ & * \text{A4\_Score} + 1.11 * \text{A5\_Score} + 1 * \text{A6\_Score} + 1.08 * \text{A7\_Score} \\ & + 0.95 * \text{A8\_Score} + 1.09 * \text{A9\_Score} + 0.75 * \text{A10\_Score} + 1.88 \\ & * \text{sum} - 3.8287 \end{aligned}$$

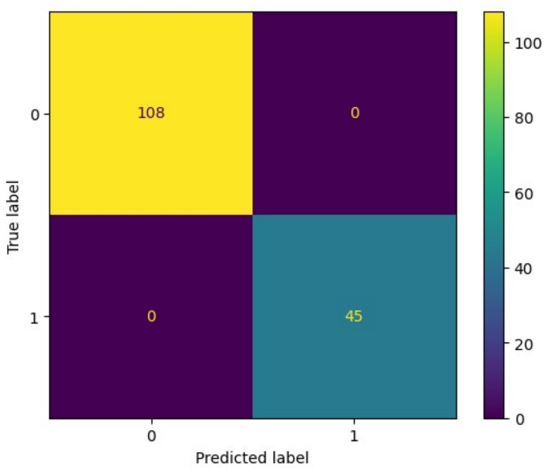
```
intercept: [-3.8287]
    Attribute  Coefficient
0    A1_Score         0.98
1    A2_Score         1.05
2    A3_Score         0.81
3    A4_Score         1.03
4    A5_Score         1.11
5    A6_Score          1.0
6    A7_Score         1.08
7    A8_Score         0.95
8    A9_Score         1.09
9   A10_Score         0.75
10        sum         1.88
```

From the coefficients of each attribute, the influence of AQ-10 scores on target variable is almost the same. Meanwhile "sum" attribute carried the heaviest weight among all the attributes, hence it gives the most effect on the target variable.

Confusion Matrix and AUC: Similar result as VotingClassifier and RF

```python
# Model perfomance
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay

cm_lg = confusion_matrix(Y_test, y_predict_rf)
lg_TP = cm_lg[0][0]
lg_FP = cm_lg[0][1]
lg_FN = cm_lg[1][0]
lg_TN = cm_lg[1][1]

print(lg.__class__.__name__,
      '\nLogistic Regression accuracy: ', round(accuracy_score(Y_test, y_predict_lg), 4),
      '\nLogistic Regression precision: ',round(precision_score(Y_test, y_predict_lg),4), # TP/(TP+FP),
      '\nLogistic Regression recall: ', round(recall_score(Y_test, y_predict_lg), 4), # TP/(TP+FN),
      '\nLogistic Regression specificity: ', round(lg_TN/(lg_TN+lg_FP), 4),
      '\nLogistic Regression misclassification: ', round(1-accuracy_score(Y_test, y_predict_lg), 4)
      )

lg_matrix = ConfusionMatrixDisplay(confusion_matrix=cm_lg, display_labels=lg.classes_)
lg_matrix.plot()

# ROC - AUC
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

lg_auc = roc_auc_score(Y_test, y_predict_lg)
print("Logistic Regression AUC: ", round(lg_auc, 4))
```

```
LogisticRegression
Logistic Regression accuracy:  1.0
Logistic Regression precision:  1.0
Logistic Regression recall:  1.0
Logistic Regression specificity:  1.0
Logistic Regression misclassification:  0.0
Logistic Regression AUC:  1.0
```



cleaned_autism_screening.csv

Before the modelling process start, a few steps are taken:

- Libraries import
- Import and read the CSV
- Convert the "Yes" and "No" values into 1 and 0
- Apply one-hot encoding to convert all string objects into numeric
- Split data into 75% training and 25% testing
- Feature scaling

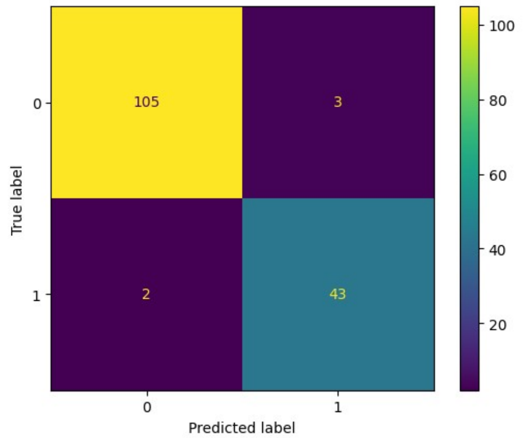
```python
# import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# import and read cleaned_autism_screening.csv
clean_df = pd.read_csv("cleaned_autism_screening.csv")

# convert "yes" and "no" values into 1 and 0
clean_df['Class/ASD']=clean_df['Class/ASD'].str.lower()
for col in ('fam_with_autism', 'used_app_before', 'Class/ASD'):
    clean_df[col]=clean_df[col].map({'no': 0, 'yes': 1})

# Applying one-hot encoding
encode_clean_df = pd.get_dummies(clean_df, dtype=int)

# split data
X = encode_clean_df.drop("Class/ASD", axis=1)
Y = encode_clean_df['Class/ASD']

X_train,X_test,Y_train,Y_test = train_test_split(
    X, Y, stratify=Y, test_size=0.25, random_state=1)

# feature scaling
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Note: the snippets for modelling and performance measure are same as the images above, hence current snippets are not displayed.

Ensemble Learning - VotingClassifier

The accuracy and AUC of VotingClassifier is close to ideal value of 1.0.

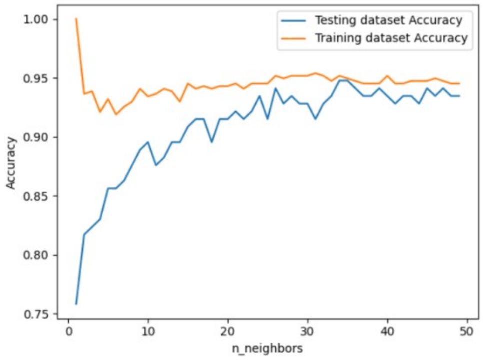
```
TP:  105
FP:  3
FN:  2
TN:  43
VotingClassifier
Ensemble accuracy:  0.9673
Ensemble precision:  0.9348
Ensemble recall:  0.9556
Ensemble specificity:  0.9348
Ensemble misclassification:  0.0327
Ensemble AUC:  0.9639
```



KNN

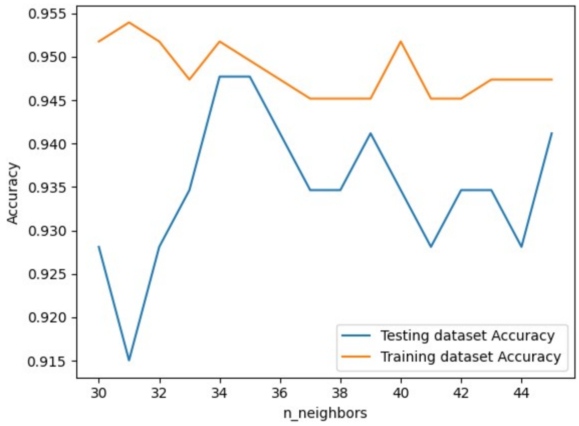
The k-value range of 1 to 50 is run to find the highest accuracy for both training and testing data. From the graph, accuracy of both data sets is closest between 30 to 45.

```python
# Trial - find out which k-value is the best
neighbors = np.arange(1, 50)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train)

    # Compute training and test data accuracy
    train_accuracy[i] = knn.score(X_train, Y_train)
    test_accuracy[i] = knn.score(X_test, Y_test)
    print(k, test_accuracy[i])

plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



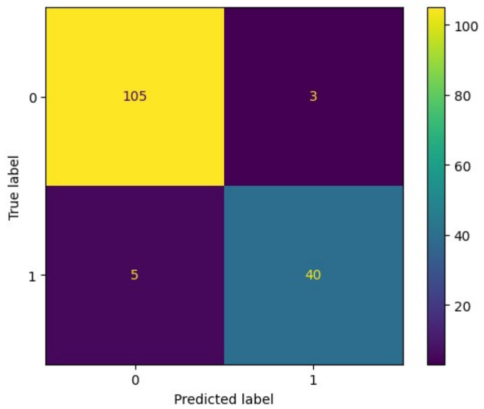
Hence, the range is set between 30 to 45: the highest accuracy of test data is at k=34 and 35, with accuracy at 0.945.

```
30 0.9281045751633987
31 0.9150326797385621
32 0.9281045751633987
33 0.934640522875817
34 0.9477124183006536
35 0.9477124183006536
36 0.9411764705882353
37 0.934640522875817
38 0.934640522875817
39 0.9411764705882353
40 0.934640522875817
41 0.9281045751633987
42 0.934640522875817
43 0.934640522875817
44 0.9281045751633987
45 0.9411764705882353
```



By setting k=35, the Confusion Matrix and AUC result are displayed as below:

```
TP:  105
FP:  3
FN:  5
TN:  40
KNeighborsClassifier
KNN accuracy:  0.9477
KNN precision:  0.9302
KNN recall:  0.8889
KNN specificity:  0.9302
KNN misclassification:  0.0523
KNN AUC:  0.9306
```

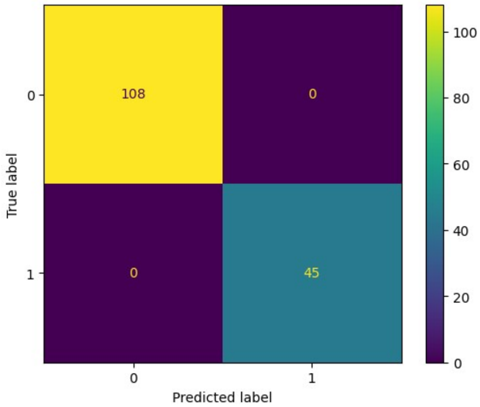


The accuracy stood firm at 0.9477, where the FP and FN did not exceed 5.

RF

This model produced the ideal result with accuracy of 1.0

```
TP:  108
FP:  0
FN:  0
TN:  45
RandomForestClassifier
Random Forest accuracy:  1.0
Random Forest precision:  1.0
Random Forest recall:  1.0
Random Forest specificity:  1.0
Random Forest misclassification:  0.0
Random Forest AUC:  1.0
```



LR

```
intercept: [-3.2002]
                     Attribute  Coefficient
13                   sum_score         1.72
5                    A6_Score          1.13
8                    A9_Score          1.12
6                    A7_Score          1.11
4                    A5_Score          1.04
..                         ...          ...
53       country_of_res_India         -0.03
37     country_of_res_Bolivia         -0.02
35  country_of_res_Bangladesh         -0.02
30     country_of_res_Armenia         -0.01
60      country_of_res_Mexico         -0.0

[92 rows x 2 columns]
```

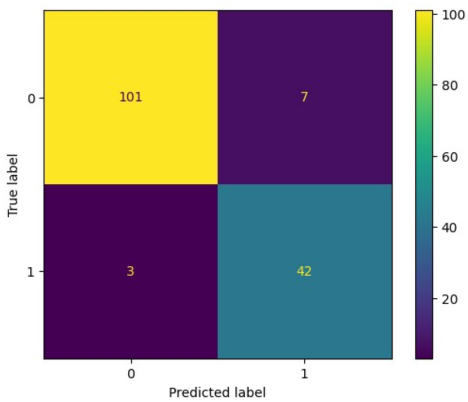
From the result, the equation of the LR is:

$$\mathrm{Class/ASD} = 1.72 * \mathrm{sum\_score} + 1.73 * \mathrm{A6\_Score} + 1.12 * \mathrm{A9\_Score} + \ldots - 0.01 * \mathrm{country\_of\_res\_Armenia} - 3.2002$$

Therefore, “sum_score” has the most ability among 92 features in affecting the target variable, “Class/ASD”; while “country_of_res_Mexico” carries the least or almost none.

For the evaluation, accuracy is 0.9346 and AUC=0.9343

```
TP:  101
FP:  7
FN:  3
TN:  42
LogisticRegression
Logistic Regression accuracy:  0.9346
Logistic Regression precision:  0.8571
Logistic Regression recall:  0.9333
Logistic Regression specificity:  0.8571
Logistic Regression misclassification:  0.0654
Logistic Regression AUC:  0.9343
```



ROC

The snippet of ROC for VotingClassifier, KNN, RF, and LR:

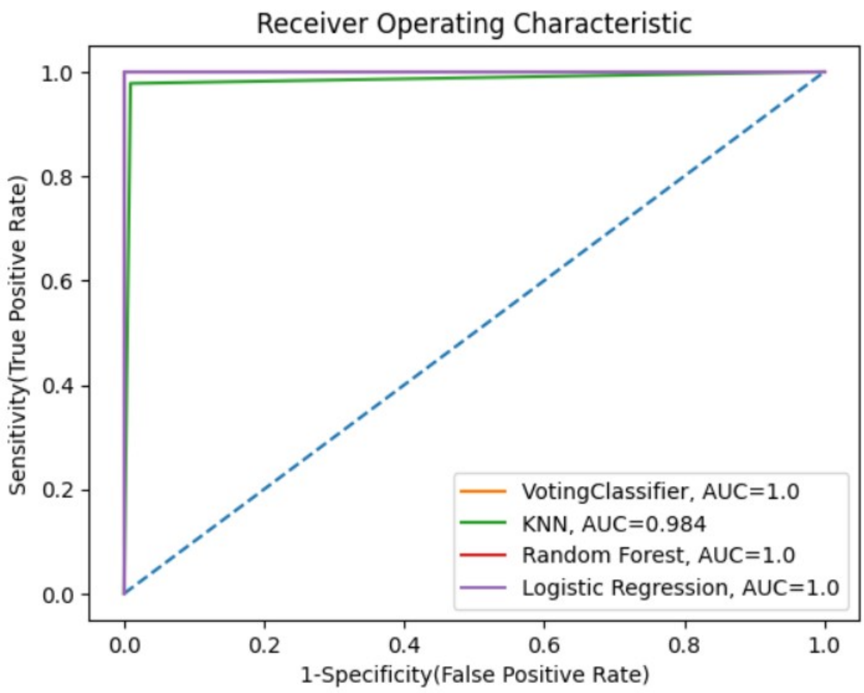
```
# apply models for predictions
y_predict_en
y_predict_knn
y_predict_rf
y_predict_lg

# derive ROC AUC scores of each model
en_auc
knn_auc
rf_auc
lg_auc

# initiate the plots of ROC charts for each model
plt.figure(0).clf()
plt.plot([0, 1], ls="--")

#fit Ensemble Learning - VotingClassifier and plot ROC curve
fpr, tpr, _ = roc_curve(Y_test, y_predict_en)
plt.plot(fpr,tpr,label="VotingClassifier, AUC="+str(round(en_auc,3)))

#fit KNN model and plot ROC curve
fpr, tpr, _ = roc_curve(Y_test, y_predict_knn)
plt.plot(fpr,tpr,label="KNN, AUC="+str(round(knn_auc,3)))

#fit Random Forest model and plot ROC curve
fpr, tpr, _ = roc_curve(Y_test, y_predict_rf)
plt.plot(fpr,tpr,label="Random Forest, AUC="+str(round(rf_auc,3)))

#fit Logistic Regression model and plot ROC curve
fpr, tpr, _ = roc_curve(Y_test, y_predict_lg)
plt.plot(fpr,tpr,label="Logistic Regression, AUC="+str(round(lg_auc,3)))

#add legend information
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

7. Model Evaluation results

Data set: score.csv

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">Metrics</td></tr>
<tr><td>Confusion Matrix</td><td>ROC-AUC</td></tr>
<tr><td rowspan="2">Models</td><td>Ensemble Learning: Voting Classifier</td><td>Accuracy = 1.0
Precision = 1.0
Recall = 1.0
Specificity = 1.0
Misclassification = 0.0</td><td>AUC = 1.0</td></tr>
<tr><td>KNN</td><td>Accuracy = 0.9869
Precision = 0.9778</td><td>AUC = 0.9843</td></tr>
</table>

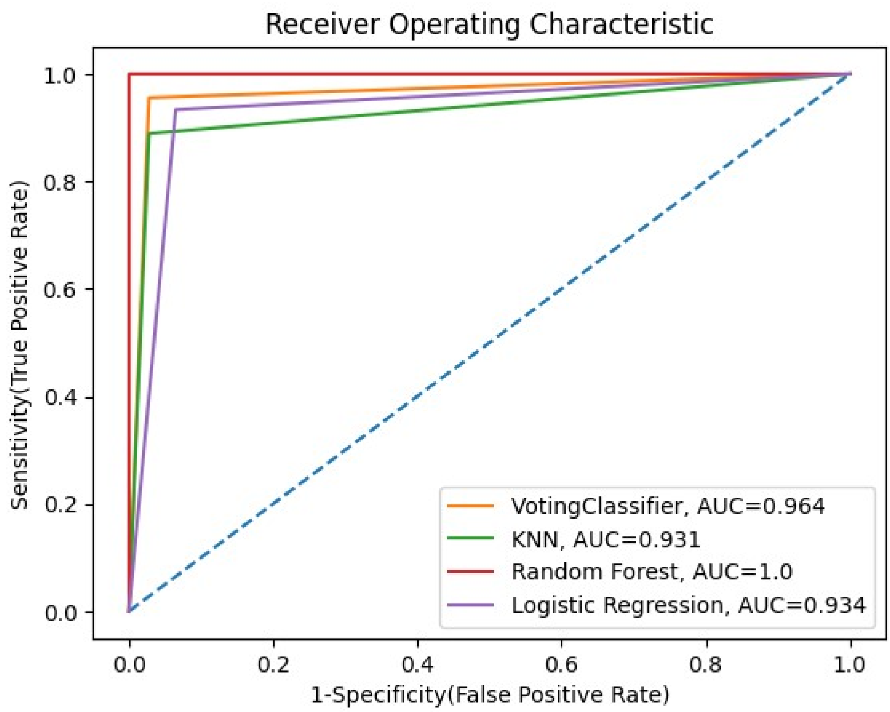
		Recall = 0.9778 Specificity = 0.9778 Misclassification = 0.0131	
	RF	Accuracy = 1.0 Precision = 1.0 Recall = 1.0 Specificity = 1.0 Misclassification = 0.0	AUC = 1.0
	LR	Accuracy = 1.0 Precision = 1.0 Recall = 1.0 Specificity = 1.0 Misclassification = 0.0	AUC = 1.0



Data set: cleaned_autism_screening.csv

		Metrics	
		Confusion Matrix	ROC-AUC
Models	Ensemble Learning	Accuracy = 0.9673 Precision = 0.9348 Recall = 0.9556	AUC = 0.9639

		Specificity = 0.9348 Misclassification = 0.0327	
	KNN	Accuracy = 0.9477 Precision = 0.9302 Recall = 0.8889 Specificity = 0.9302 Misclassification = 0.0523	AUC = 0.9306
	RF	Accuracy = 1.0 Precision = 1.0 Recall = 1.0 Specificity = 1.0 Misclassification = 1.0	AUC = 1.0
	LR	Accuracy = 0.9346 Precision = 0.8571 Recall = 0.9333 Specificity = 0.8571 Misclassification = 0.0654	AUC = 0.9343



Conclusion

score.csv

From the ROC, since KNN has the lowest AUC value although the value is close to 1.0, while VotingClassifer, LR and RF produce the same confusion matrix and ROC result of 1.0. Therefore, KNN is the least preferred model compared to the other 3.

cleaned_autism_screening.csv

The shape of ROC curve suggests RF as the best model due to the highest accuracy, AUC value, followed by VotingClassifier, LR and lastly KNN.

(2654 words)

Reference

Larxel. (2017). *Autism Screening on Adults*. Kaggle.com. https://www.kaggle.com/datasets/andrewmvd/autism-screening-on-adults?resource=download

UCI Machine Learning Repository. (2017). Uci.edu. https://archive.ics.uci.edu/dataset/426/autism+screening+adult

Afarin Bargrizan. (2023). *ASD questionnairs- Final*. Kaggle.com. https://www.kaggle.com/datasets/afarinbargrizan/asd-final

Engelbrecht, N. (2020, April 20). *The AQ-10*. Embrace Autism. https://embrace-autism.com/aq-10/#The_AQ-10

Nurul Amirah Mashudi, Ahmad, N., & Norliza Mohd Noor. (2021). Classification of adult autistic spectrum disorder using machine learning approach. *IAES International Journal of Artificial Intelligence*, *10*(3), 743–743. https://doi.org/10.11591/ijai.v10.i3.pp743-751

Autism spectrum disorder (ASD) | Autism Speaks. (2024). Autism Speaks. https://www.autismspeaks.org/what-autism

Pierce, K., Carter, C., Weinfeld, M., Desmond, J., Hazin, R., Bjork, R., & Gallagher, N. (2011). Detecting, Studying, and Treating Autism Early: The One-Year Well-

Baby Check-Up Approach. *The Journal of Pediatrics*, *159*(3), 458-465.e6. https://doi.org/10.1016/j.jpeds.2011.02.036

When do children usually show symptoms of autism? (2017, January 31). Https://Www.nichd.nih.gov/. https://www.nichd.nih.gov/health/topics/autism/conditioninfo/symptoms-appear#f4

NHS Choices. (2024). *Signs of autism in adults*. https://www.nhs.uk/conditions/autism/signs/adults/

How is Autism Different in Women? (2023). Harvard.edu. https://adult-autism.health.harvard.edu/resources/how-is-autism-different-in-females/

Fadi. (2018). *Autism screening data for toddlers*. Kaggle.com. https://www.kaggle.com/datasets/fabdelja/autism-screening-for-toddlers

ASD. (2017). Asdtests.com. https://www.asdtests.com/

Allison, C., Auyeung, B., & Baron-Cohen, S. (2011). Toward Brief "Red Flags" for Autism Screening: The Short Autism Spectrum Quotient and the Short Quantitative Checklist in 1,000 Cases and 3,000 Controls. *Journal of the American Academy of Child & Adolescent Psychiatry*, *51*(2), 202-212.e7. https://doi.org/10.1016/j.jaac.2011.11.003

Umar, M. (2024). *Autism Spectrum*. Kaggle.com. https://www.kaggle.com/datasets/umeradnaan/autism-screening

BaggingClassifier. (2024). Scikit-Learn. https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.BaggingClassifier.html

Simplilearn. (2021, March 26). *Ensemble Learning: From Basics to Advanced Techniques!* Simplilearn.com; Simplilearn. https://www.simplilearn.com/ensemble-learning-article

LinkedIn. (2024). Linkedin.com. https://www.linkedin.com/pulse/bagging-classifier-rupak-roy/

GeeksforGeeks. (2019, April 9). *knearest neighbor algorithm in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/

SoulPage IT Solutions. (2023, June 19). Soulpage IT Solutions. https://soulpageit.com/ai-glossary/ensemble-votingexplained/#:~:text=Improved%20Accuracy%3A%20It%20leverages%20the,and%20achieve%20better%20overall%20performance.

AIML.com. (2022, June 11). *What are the advantages and disadvantages of Random Forest?* AIML.com. https://aiml.com/what-are-the-advantages-and-disadvantages-of-random-forest/

Accuracy vs. precision vs. recall in machine learning: what's the difference? (2024). Evidentlyai.com. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Recall%20can%20also%20be%20called,research%20rather%20than%20machine%20learning.

Verma, A. (2023, January 27). *Evaluation Metrics for Classification and Regression: A Comprehensive Guide*. DEV Community. https://dev.to/anurag629/evaluation-metrics-for-classification-and-regression-a-comprehensive-guide-47hb#:~:text=The%20mean%20squared%20error%20is,the%20performance%20of%20regression%20models.

Christos Goumopoulos, & Stergiopoulos, N. G. (2022). Mental stress detection using a wearable device and heart rate variability monitoring. *Elsevier EBooks*, 261–290. https://doi.org/10.1016/b978-0-323-90585-5.00011-4

Dr Ana Rojo-Echeburúa. (2024, June 26). *What Is One Hot Encoding and How to Implement It in Python*. Datacamp.com; DataCamp. https://www.datacamp.com/tutorial/one-hot-encoding-python-tutorial?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=157098106775&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=716160943678&utm_targetid=aud-1704732079567:dsa-2264919291989&utm_loc_interest_ms=&utm_loc_physical_ms=9066731&utm_content=&utm_campaign=230119_1-sea~dsa~tofu_2-b2c_3-row-p2_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na-oct24&gad_source=1&gclid=CjwKCAjw3624BhBAEiwAkxgTOmITa6qWv2vhmu3rpLNb-N0SLydR69zq_Qbl51HOq87mBga8ESM1JRoCMDMQAvD_BwE

GeeksforGeeks. (2019, June 12). *One Hot Encoding in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-one-hot-encoding

Syed Wahaj. (2023, August 8). *Hard vs. Soft Voting Classifiers - Techkluster*. Techkluster. https://techkluster.com/technology/hard-vs-soft-voting-classifiers/#:~:text=Disadvantages%3A,be%20affected%20by%20outlier%20predictions.