Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import plotly.figure_factory as ff
import plotly.express as px
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn import metrics
from sklearn.impute import KNNImputer
from sklearn import preprocessing
from numpy import isnan
import pickle
```

```
from keras.models import Sequential
from keras.layers import Dense, Conv1D
from keras.layers import Dense, Flatten, Conv1D, MaxPool1D, Dropout
import keras
from keras.utils import to_categorical
```

Importing Dataset

```
data = pd.read_csv("Autism-Adult-Data.csv")
```

EDA

```
data.head()
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score
0	1	1	1	1	1	0	0	1	1
1	2	1	1	0	1	0	0	0	1
2	3	1	1	0	1	1	0	1	1
3	4	1	1	0	1	0	0	1	1
4	5	1	0	0	0	0	0	0	1

5 rows × 22 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               704 non-null    int64 
 1   A1_Score         704 non-null    int64 
 2   A2_Score         704 non-null    int64 
 3   A3_Score         704 non-null    int64 
 4   A4_Score         704 non-null    int64 
 5   A5_Score         704 non-null    int64 
 6   A6_Score         704 non-null    int64 
 7   A7_Score         704 non-null    int64 
 8   A8_Score         704 non-null    int64 
 9   A9_Score         704 non-null    int64 
 10  A10_Score        704 non-null    int64 
 11  age              704 non-null    object
 12  gender           704 non-null    object
 13  ethnicity        704 non-null    object
 14  jundice          704 non-null    object
 15  austim           704 non-null    object
 16  contry_of_res    704 non-null    object
 17  used_app_before  704 non-null    object
 18  result           704 non-null    int64 
 19  age_desc         704 non-null    object
 20  relation         704 non-null    object
 21  Class/ASD        704 non-null    object
dtypes: int64(12), object(10)
memory usage: 121.1+ KB
```

```
data.isnull().sum()
```

```
id                0
A1_Score          0
A2_Score          0
A3_Score          0
A4_Score          0
A5_Score          0
A6_Score          0
A7_Score          0
A8_Score          0
A9_Score          0
A10_Score         0
age               0
gender            0
ethnicity         0
jundice           0
austim            0
contry_of_res     0
used_app_before   0
result            0
age_desc          0
relation          0
Class/ASD         0
dtype: int64
```

```
df = data.replace('?', np.nan)
```

```
df.isnull().sum()
```

```
id                 0
A1_Score           0
```

```
A2_Score            0
A3_Score            0
A4_Score            0
A5_Score            0
A6_Score            0
A7_Score            0
A8_Score            0
A9_Score            0
A10_Score           0
age                 2
gender              0
ethnicity          95
jundice             0
austim              0
contry_of_res       0
used_app_before     0
result              0
age_desc            0
relation           95
Class/ASD           0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               704 non-null    int64 
 1   A1_Score         704 non-null    int64 
 2   A2_Score         704 non-null    int64 
 3   A3_Score         704 non-null    int64 
 4   A4_Score         704 non-null    int64 
 5   A5_Score         704 non-null    int64 
 6   A6_Score         704 non-null    int64 
 7   A7_Score         704 non-null    int64 
 8   A8_Score         704 non-null    int64 
 9   A9_Score         704 non-null    int64 
 10  A10_Score        704 non-null    int64 
 11  age              702 non-null    object
 12  gender           704 non-null    object
 13  ethnicity        609 non-null    object
 14  jundice          704 non-null    object
 15  austim           704 non-null    object
 16  contry_of_res    704 non-null    object
 17  used_app_before  704 non-null    object
 18  result           704 non-null    int64 
 19  age_desc         704 non-null    object
 20  relation         609 non-null    object
 21  Class/ASD        704 non-null    object
dtypes: int64(12), object(10)
memory usage: 121.1+ KB
```

Handling categorical data

```
le = preprocessing.LabelEncoder()
```

```
df = df.dropna()
```

```
df['age'] = le.fit_transform(df['age'])
df['gender'] = le.fit_transform(df['gender'])
df['ethnicity'] = le.fit_transform(df['ethnicity'])
df['jundice'] = le.fit_transform(df['jundice'])
df['austim'] = le.fit_transform(df['austim'])
df['contry_of_res'] = le.fit_transform(df['contry_of_res'])
df['used_app_before'] = le.fit_transform(df['used_app_before'])
df['age_desc'] = le.fit_transform(df['age_desc'])
df['relation'] = le.fit_transform(df['relation'])
df['Class/ASD'] = le.fit_transform(df['Class/ASD'])
```

```
df
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Sc
0	1	1	1	1	1	0	0	1	
1	2	1	1	0	1	0	0	0	
2	3	1	1	0	1	1	0	1	
3	4	1	1	0	1	0	0	1	
5	6	1	1	1	1	1	0	1	
...	...	...	...	...	...	...	...	...	
698	699	1	1	1	1	1	1	1	
699	700	0	1	0	1	1	0	1	
700	701	1	0	0	0	0	0	0	
702	703	1	0	0	1	1	0	1	
703	704	1	0	1	1	1	0	1	

609 rows × 22 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 609 entries, 0 to 703
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               609 non-null    int64
 1   A1_Score         609 non-null    int64
 2   A2_Score         609 non-null    int64
 3   A3_Score         609 non-null    int64
 4   A4_Score         609 non-null    int64
 5   A5_Score         609 non-null    int64
 6   A6_Score         609 non-null    int64
 7   A7_Score         609 non-null    int64
 8   A8_Score         609 non-null    int64
 9   A9_Score         609 non-null    int64
 10  A10_Score        609 non-null    int64
 11  age              609 non-null    int64
 12  gender           609 non-null    int64
 13  ethnicity        609 non-null    int64
 14  jundice          609 non-null    int64
 15  austim           609 non-null    int64
 16  contry_of_res    609 non-null    int64
 17  used_app_before  609 non-null    int64
 18  result           609 non-null    int64
```

```
 19  age_desc         609 non-null    int64
 20  relation         609 non-null    int64
 21  Class/ASD        609 non-null    int64
dtypes: int64(22)
memory usage: 109.4 KB
```

KNN IMPUTATION

```
X = df.drop(['Class/ASD'], axis=1)
```

```
y = df['Class/ASD']
```

```
X
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Sc
0	1	1	1	1	1	0	0	1	
1	2	1	1	0	1	0	0	0	
2	3	1	1	0	1	1	0	1	
3	4	1	1	0	1	0	0	1	
5	6	1	1	1	1	1	0	1	
...	...	...	...	...	...	...	...	...	...
698	699	1	1	1	1	1	1	1	
699	700	0	1	0	1	1	0	1	
700	701	1	0	0	0	0	0	0	
702	703	1	0	0	1	1	0	1	
703	704	1	0	1	1	1	0	1	

609 rows × 21 columns

```
imputer = KNNImputer()
```

```
imputer.fit(X)
```

```
KNNImputer()
```

```
Xtrans = imputer.transform(X)
```

```
Xtrans
```

```
array([[  1.,   1.,   1., ...,   6.,   0.,   4.],
       [  2.,   1.,   1., ...,   5.,   0.,   4.],
       [  3.,   1.,   1., ...,   8.,   0.,   2.],
       ...,
       [701.,   1.,   0., ...,   3.,   0.,   2.],
       [703.,   1.,   0., ...,   6.,   0.,   4.],
       [704.,   1.,   0., ...,   8.,   0.,   4.]])
```

```
print('Missing: %d' % sum(isnan(Xtrans).flatten()))
```

```
Missing: 0
```

Model Building

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
X_train
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Sc
587	588	1	1	1	1	1	1	0	
103	104	1	1	0	0	1	0	1	
614	615	1	1	0	1	0	0	0	
521	522	1	1	1	1	1	0	0	
213	214	1	0	1	1	0	0	0	
...	...	...	...	...	...	...	...	...	
298	299	1	0	0	1	0	0	1	
10	11	1	1	1	1	1	1	1	
425	426	1	0	0	0	0	0	1	
205	206	1	1	1	1	1	1	1	
648	649	1	0	1	1	1	0	0	

487 rows × 21 columns

SVM

```
from sklearn.svm import LinearSVC
SVM = LinearSVC(random_state=0, tol=1e-5)
SVM.fit(X_train, y_train)
predictions = SVM.predict(X_test)
val1 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for SVM: ", val1, "\n")
print("*Confusion Matrix for SVM: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for SVM: ")
print(classification_report(y_test, predictions))
```

```
*Accuracy score for SVM:  90.98360655737704

*Confusion Matrix for SVM:
[[82  6]
 [ 5 29]]
*Classification Report for SVM:
              precision    recall  f1-score   support

           0       0.94      0.93      0.94        88
```

```
                1       0.83      0.85      0.84        34

         accuracy                           0.91       122
        macro avg       0.89      0.89      0.89       122
     weighted avg       0.91      0.91      0.91       122
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
GNB.fit(X_train, y_train)
predictions = GNB.predict(X_test)
val2 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for GNB: ", val2, "\n")
print("*Confusion Matrix for GNB: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for GNB: ")
print(classification_report(y_test, predictions))
```

```
     *Accuracy score for GNB:  98.36065573770492

     *Confusion Matrix for GNB:
     [[87  1]
      [ 1 33]]
     *Classification Report for GNB:
                   precision    recall  f1-score   support

                0       0.99      0.99      0.99        88
                1       0.97      0.97      0.97        34

         accuracy                           0.98       122
        macro avg       0.98      0.98      0.98       122
     weighted avg       0.98      0.98      0.98       122
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(X_train, y_train)
predictions = LR.predict(X_test)
val3 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for LR: ", val3, "\n")
print("*Confusion Matrix for LR: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for LR: ")
print(classification_report(y_test, predictions))
```

```
     *Accuracy score for LR:  97.54098360655738

     *Confusion Matrix for LR:
     [[85  3]
      [ 0 34]]
     *Classification Report for LR:
                   precision    recall  f1-score   support

                0       1.00      0.97      0.98        88
```

```
                1       0.92      1.00      0.96        34

         accuracy                           0.98       122
        macro avg       0.96      0.98      0.97       122
     weighted avg       0.98      0.98      0.98       122
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier()
KNN.fit(X_train, y_train)
predictions = KNN.predict(X_test)
val4 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for KNN: ", val4, "\n")
print("*Confusion Matrix for KNN: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for KNN: ")
print(classification_report(y_test, predictions))
```

```
     *Accuracy score for KNN:  72.1311475409836

     *Confusion Matrix for KNN:
     [[75 13]
      [21 13]]
     *Classification Report for KNN:
                   precision    recall  f1-score   support

                0       0.78      0.85      0.82        88
                1       0.50      0.38      0.43        34

         accuracy                           0.72       122
        macro avg       0.64      0.62      0.62       122
     weighted avg       0.70      0.72      0.71       122
```

MLP

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(random_state=1, max_iter=300)
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
val5 = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for MLP: ", val5, "\n")
print("*Confusion Matrix for MLP: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for MLP: ")
print(classification_report(y_test, predictions))
```

```
     *Accuracy score for MLP:  95.08196721311475

     *Confusion Matrix for MLP:
     [[84  4]
      [ 2 32]]
     *Classification Report for MLP:
                   precision    recall  f1-score   support

                0       0.98      0.95      0.97        88
```

```
                1       0.89      0.94      0.91        34

         accuracy                           0.95       122
        macro avg       0.93      0.95      0.94       122
     weighted avg       0.95      0.95      0.95       122
```

```
X_train = X_train.values
X_test = X_test.values
```

```
X_train = X_train.reshape(-1, X_train.shape[1],1)
X_test = X_test.reshape(-1, X_test.shape[1],1)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(487, 21, 1)
(122, 21, 1)
```

```
print(y_train.shape)
print(y_test.shape)
```

```
(487,)
(122,)
```

```
y_test
```

```
670    1
60     1
617    0
410    0
64     1
      ..
562    0
316    0
595    0
299    1
178    0
Name: Class/ASD, Length: 122, dtype: int32
```

CNN

```
model = Sequential()
model.add(Conv1D(filters=128, kernel_size=2, input_shape=(X_train.shape[1],X_train.shape[2]), activa
model.add(MaxPool1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))  # dropout
model.add(Dense(3, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
conv1d (Conv1D)              (None, 20, 128)           384
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 10, 128)           0
_________________________________________________________________
flatten (Flatten)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 64)                81984
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 82,563
Trainable params: 82,563
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(X_train, y_train, epochs = 20, batch_size = 128, validation_data=(X_test, y_test))
cnn = model.evaluate(X_test,y_test)
#print(results)
#text.insert(END, "%Accuracy of CNN Model: "+str(cnn[1]*100)+"\n")
results = []
results.append(cnn[1]*100)
print(results)
```

```
Epoch 1/20
4/4 [==============================] - 0s 106ms/step - loss: 3.5415 - accuracy: 0.4004 - val_los
Epoch 2/20
4/4 [==============================] - 0s 16ms/step - loss: 2.5578 - accuracy: 0.4127 - val_loss
Epoch 3/20
4/4 [==============================] - 0s 15ms/step - loss: 2.1558 - accuracy: 0.2916 - val_loss
Epoch 4/20
4/4 [==============================] - 0s 15ms/step - loss: 1.7554 - accuracy: 0.3655 - val_loss
Epoch 5/20
4/4 [==============================] - 0s 15ms/step - loss: 1.4256 - accuracy: 0.3450 - val_loss
Epoch 6/20
4/4 [==============================] - 0s 18ms/step - loss: 1.0072 - accuracy: 0.3799 - val_loss
Epoch 7/20
4/4 [==============================] - 0s 17ms/step - loss: 0.7468 - accuracy: 0.4292 - val_loss
Epoch 8/20
4/4 [==============================] - 0s 14ms/step - loss: 0.6599 - accuracy: 0.3142 - val_loss
Epoch 9/20
4/4 [==============================] - 0s 15ms/step - loss: 0.6264 - accuracy: 0.3388 - val_loss
Epoch 10/20
4/4 [==============================] - 0s 15ms/step - loss: 0.6160 - accuracy: 0.2156 - val_loss
Epoch 11/20
4/4 [==============================] - 0s 16ms/step - loss: 0.6142 - accuracy: 0.1314 - val_loss
Epoch 12/20
4/4 [==============================] - 0s 17ms/step - loss: 0.6137 - accuracy: 0.0965 - val_loss
Epoch 13/20
4/4 [==============================] - 0s 15ms/step - loss: 0.6136 - accuracy: 0.0657 - val_loss
Epoch 14/20
4/4 [==============================] - 0s 16ms/step - loss: 0.6134 - accuracy: 0.0493 - val_loss
Epoch 15/20
4/4 [==============================] - 0s 16ms/step - loss: 0.6135 - accuracy: 0.0370 - val_loss
Epoch 16/20
4/4 [==============================] - 0s 18ms/step - loss: 0.6134 - accuracy: 0.0267 - val_loss
Epoch 17/20
4/4 [==============================] - 0s 18ms/step - loss: 0.6134 - accuracy: 0.0226 - val_loss
Epoch 18/20
4/4 [==============================] - 0s 16ms/step - loss: 0.6134 - accuracy: 0.0185 - val_loss
Epoch 19/20
4/4 [==============================] - 0s 15ms/step - loss: 0.6134 - accuracy: 0.0246 - val_loss
Epoch 20/20
4/4 [==============================] - 0s 17ms/step - loss: 0.6133 - accuracy: 0.0164 - val_loss
4/4 [==============================] - 0s 24ms/step - loss: 0.5987 - accuracy: 0.0574
```

```
[5.737704783678055]
```

```
X_train.shape
```

```
(487, 21, 1)
```

```
y_train.shape
```

```
(487,)
```

Pickle

```
print(df.corr()["Class/ASD"].abs().sort_values(ascending=False))
```

```
Class/ASD          1.000000
result             0.826767
A9_Score           0.641386
A6_Score           0.606399
A5_Score           0.550717
A4_Score           0.470414
A3_Score           0.435016
A10_Score          0.377310
A7_Score           0.355280
A2_Score           0.335059
A1_Score           0.293088
A8_Score           0.246237
ethnicity          0.225384
austim             0.164870
age                0.130729
jundice            0.128491
gender             0.085606
used_app_before    0.057888
relation           0.036138
id                 0.033500
contry_of_res      0.001081
age_desc                NaN
Name: Class/ASD, dtype: float64
```

```
X = df.drop(['ethnicity','austim','age','jundice','gender','used_app_before','relation','id','contry
```

```
X
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score
0	1	1	1	1	0	0	1	1
1	1	1	0	1	0	0	0	1
2	1	1	0	1	1	0	1	1
3	1	1	0	1	0	0	1	1
5	1	1	1	1	1	0	1	1
...	...	...	...	...	...	...	...	...
698	1	1	1	1	1	1	1	1
699	0	1	0	1	1	0	1	1
700	1	0	0	0	0	0	0	1
702	1	0	0	1	1	0	1	0
703	1	0	1	1	1	0	1	1

609 rows × 11 columns

```
y = df['Class/ASD']
```

```
y
```

```
0      0
1      0
2      1
3      0
5      1
      ..
698    1
699    1
700    0
702    0
703    1
Name: Class/ASD, Length: 609, dtype: int32
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
GNB.fit(X_train, y_train)
predictions = GNB.predict(X_test)
val = (accuracy_score(y_test, predictions)*100)
print("*Accuracy score for GNB: ", val, "\n")
print("*Confusion Matrix for GNB: ")
print(confusion_matrix(y_test, predictions))
print("*Classification Report for GNB: ")
print(classification_report(y_test, predictions))
```

```
*Accuracy score for GNB:  99.18032786885246

*Confusion Matrix for GNB:
[[87  1]
 [ 0 34]]
*Classification Report for GNB:
              precision    recall  f1-score   support
```

```
                0       1.00      0.99      0.99        88
                1       0.97      1.00      0.99        34

         accuracy                           0.99       122
        macro avg       0.99      0.99      0.99       122
     weighted avg       0.99      0.99      0.99       122
```

```
pickle.dump(GNB, open('model.pkl', 'wb'))
```

```
s = np.array([0,1,1,1,1,1,1,1,1,7])
print(s.shape)
s = s.reshape(1,-1)
print(s.shape)
```

```
(11,)
(1, 11)
```

```
model = pickle.load(open('model.pkl', 'rb'))
print(model.predict(s))
```

```
[1]
```

Start coding or generate with AI.