In [146…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [147…
```
df = pd.read_csv('data.csv',sep='\t')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39775 entries, 0 to 39774
Columns: 172 entries, Q1A to major
dtypes: int64(170), object(2)
memory usage: 52.2+ MB
```

Data Cleaning

In [148…
```
df['wrongansw'] = 0
df['wrongansw'] = df['wrongansw'].where(df['VCL6']== 0, df['wrongansw'] + 1)
df['wrongansw'] = df['wrongansw'].where(df['VCL9']== 0, df['wrongansw'] + 1)
df['wrongansw'] = df['wrongansw'].where(df['VCL12']== 0, df['wrongansw'] + 1)

df.wrongansw.value_counts()
```

Out[148…
```
wrongansw
0    34583
1     4044
2      822
3      326
Name: count, dtype: int64
```

In [149…
```
df = df[df['wrongansw'].isin([2, 3])]
df = df.drop(columns='wrongansw')
df.head(3)
df.shape
```

Out[149…
```
(1148, 172)
```

In [150…
```
vcls = []
for i in range(1, 17):
    vcls.append('VCL' + str(i))
```

```
df = df.drop(columns=vcls)
df.shape
```

Out[150… (1148, 156)

In [151…
```
categorical = df.select_dtypes('object').columns

print('Categorical Columns: ', df[categorical].columns)
print(df[categorical].nunique())

df = df.drop(['major'], axis=1)
```

```
Categorical Columns:  Index(['country', 'major'], dtype='object')
country     68
major      365
dtype: int64
```

Labelling the questions that are for specific mental disorder

In [152…
```
DASS_keys = {
    'Depression': [3, 5, 10, 13, 16, 17, 21, 24, 26, 31, 34, 37, 38, 42],
    'Anxiety': [2, 4, 7, 9, 15, 19, 20, 23, 25, 28, 30, 36, 40, 41],
    'Stress': [1, 6, 8, 11, 12, 14, 18, 22, 27, 29, 32, 33, 35, 39]
    }
```

Data Collection

In [153…
```
depr = []
for i in DASS_keys["Depression"]:
    depr.append('Q' + str(i) + 'A')

anx = []
for i in DASS_keys["Anxiety"]:
    anx.append('Q' + str(i) + 'A')

stre = []
for i in DASS_keys["Stress"]:
    stre.append('Q' + str(i) + 'A')


df_depr=df.filter(depr)
df_anx=df.filter(anx)
df_stre=df.filter(stre)
```

In [154…
```
disorders = [depr, anx, stre]

def scores(df):
    df["ScoresDepr"] = df[depr].sum(axis=1)
    df["ScoresAnx"] = df[anx].sum(axis=1)
    df["ScoresStre"] = df[stre].sum(axis=1)
```

```
    return df

for i in disorders:
        df[i] -= 1

df = scores(df)

df.head()
```

Out[154…

	Q1A	Q1I	Q1E	Q2A	Q2I	Q2E	Q3A	Q3I	Q3E	Q4A	…	hand	religion	orienta
26	3	1	4820	3	39	1109	3	8	1833	3	…	1	7	
31	1	25	3671	1	26	5743	1	13	3154	0	…	1	2	
57	1	27	3827	2	33	6293	1	22	12153	2	…	1	1	
81	3	14	2205	0	25	3320	1	40	2803	0	…	1	4	
87	2	4	8405	1	29	5819	3	32	1925	3	…	1	1	

5 rows × 158 columns

In [155…

```
# CATEGORY: DEPRESSION
CategoryDepr=[]

for i in df['ScoresDepr']:
    if i in range(0,10):
        CategoryDepr.append('Normal')
    elif i in range(10,14):
        CategoryDepr.append('Minimal')
    elif i in range(14,21):
        CategoryDepr.append('Moderate')
    elif i in range(21,28):
        CategoryDepr.append('Severe')
    else:
        CategoryDepr.append('Extremely Severe')

df['CategoryDepr']= CategoryDepr
```

In [156…

```
# CATEGORY: ANXIETY
CategoryAnx=[]

for i in df['ScoresAnx']:
    if i in range(0,8):
        CategoryAnx.append('Normal')
    elif i in range(8,10):
        CategoryAnx.append('Minimal')
    elif i in range(10,15):
        CategoryAnx.append('Moderate')
    elif i in range(15,20):
        CategoryAnx.append('Severe')
    else:
        CategoryAnx.append('Extremely Severe')
```

```
df['CategoryAnx']= CategoryAnx
```

In [157…

```
# CATEGORY: STRESS
CategoryStre=[]

for i in df['ScoresStre']:
    if i in range(0,15):
        CategoryStre.append('Normal')
    elif i in range(15,19):
        CategoryStre.append('Minimal')
    elif i in range(19,26):
        CategoryStre.append('Moderate')
    elif i in range(26,34):
        CategoryStre.append('Severe')
    else:
        CategoryStre.append('Extremely Severe')

df['CategoryStre']= CategoryStre
```

In [158…

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

Out[158…

```
0
```

In [159…

```
df
```

Out[159…

	Q1A	Q1I	Q1E	Q2A	Q2I	Q2E	Q3A	Q3I	Q3E	Q4A	…	race	voted	marri
26	3	1	4820	3	39	1109	3	8	1833	3	…	60	2	
31	1	25	3671	1	26	5743	1	13	3154	0	…	60	2	
57	1	27	3827	2	33	6293	1	22	12153	2	…	60	1	
81	3	14	2205	0	25	3320	1	40	2803	0	…	70	2	
87	2	4	8405	1	29	5819	3	32	1925	3	…	30	2	
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
39621	2	10	3293	0	16	2830	2	12	3594	2	…	70	1	
39643	1	18	3731	0	10	1623	2	13	2876	0	…	10	1	
39644	0	25	412	0	2	11824	0	29	349	0	…	30	1	
39734	3	37	3099	1	21	4454	3	1	11688	0	…	60	1	
39750	2	15	2892	3	24	1848	3	12	2395	3	…	10	2	

1148 rows × 161 columns

Data Classification Models for Depression

Data Pre-processing

In [160…

```
Y = df['CategoryDepr']
X = df.drop(columns=['Q2A','Q4A','Q7A','Q9A','Q15A','Q19A','Q20A','Q23A','Q25A','Q2
X.head()
```

Out[160…

	source	TIPI1	TIPI2	TIPI3	TIPI4	TIPI5	TIPI6	TIPI7	TIPI8	TIPI9	…	gender	age
26	2	4	1	7	7	7	2	6	5	1	…	2	15
31	2	5	5	5	5	7	7	7	5	5	…	1	18
57	0	2	4	1	5	6	4	3	6	6	…	2	21
81	2	7	6	3	6	4	1	7	5	3	…	2	14
87	2	1	3	5	6	6	7	2	7	1	…	1	17

5 rows × 23 columns

In [161…
```
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

In [162…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_st
```

Classification Models For Depression

In [163…
```
# KNN Model for Depression Classification

knn = KNeighborsClassifier(n_neighbors=95)


knn.fit(X_train, Y_train)


Y_pred = knn.predict(X_test)


accuracy_knn = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of KNN Model - Depression: {:.5f}'.format(accuracy_knn))
print('Confusion Matrix of KNN Model - Depression: \n', conf_matrix)
print('Confusion Matrix of KNN Model - Depression: \n', classif_report)
```

```
Accuracy of KNN Model - Depression: 0.59930
Confusion Matrix of KNN Model - Depression:
 [[108   0   0   7   1]
 [  2   0   1  13   0]
 [ 29   0   2   7   0]
 [  9   0   5  62   0]
 [ 36   0   1   4   0]]
Confusion Matrix of KNN Model - Depression:
                   precision    recall  f1-score   support

Extremely Severe       0.59      0.93      0.72       116
         Minimal       0.00      0.00      0.00        16
        Moderate       0.22      0.05      0.09        38
          Normal       0.67      0.82      0.73        76
          Severe       0.00      0.00      0.00        41

        accuracy                           0.60       287
       macro avg       0.30      0.36      0.31       287
    weighted avg       0.44      0.60      0.50       287
```

In [164…
```
# SVC Model for Depression Classification
svm_model = SVC(kernel='linear')

svm_model.fit(X_train, Y_train)
```

```
preds = svm_model.predict(X_test)


accuracy_svc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of SVM Model - Depression: {:.5f}'.format(accuracy_svc))
print('Confusion Matrix of SVM Model - Depression: \n', conf_matrix)
print('Confusion Matrix of SVM Model - Depression: \n', classif_report)
```

```
Accuracy of SVM Model - Depression: 0.59930
Confusion Matrix of SVM Model - Depression:
 [[108   0   0   7   1]
 [  2   0   1  13   0]
 [ 29   0   2   7   0]
 [  9   0   5  62   0]
 [ 36   0   1   4   0]]
Confusion Matrix of SVM Model - Depression:
                  precision    recall  f1-score   support

Extremely Severe       0.59      0.93      0.72       116
         Minimal       0.00      0.00      0.00        16
        Moderate       0.22      0.05      0.09        38
          Normal       0.67      0.82      0.73        76
          Severe       0.00      0.00      0.00        41

        accuracy                           0.60       287
       macro avg       0.30      0.36      0.31       287
    weighted avg       0.44      0.60      0.50       287
```

In [165…

```
# Naive Bayes Model for Depression Classification

clfNB = GaussianNB()


clfNB.fit(X_train, Y_train)


Y_pred = clfNB.predict(X_test)


accuracy_nb = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of NB Model - Depression: {:.5f}'.format(accuracy_nb))
print('Confusion Matrix of NB Model - Depression: \n', conf_matrix)
print('Classification Report of NB Model - Depression: \n', classif_report)
```

```
Accuracy of NB Model - Depression: 0.94425
Confusion Matrix of NB Model - Depression:
 [[113   0   0   0   3]
 [  0  12   1   3   0]
 [  0   0  37   0   1]
 [  0   5   0  71   0]
 [  2   0   1   0  38]]
Classification Report of NB Model - Depression:
                   precision    recall  f1-score   support

Extremely Severe       0.98      0.97      0.98       116
         Minimal       0.71      0.75      0.73        16
        Moderate       0.95      0.97      0.96        38
          Normal       0.96      0.93      0.95        76
          Severe       0.90      0.93      0.92        41

        accuracy                           0.94       287
       macro avg       0.90      0.91      0.91       287
    weighted avg       0.95      0.94      0.94       287
```

In [166…

```python
# Random Forest Classifier Model for Depression Classification

clfRFC = RandomForestClassifier(n_estimators=110, random_state=110)


clfRFC.fit(X_train, Y_train)


Y_pred = clfRFC.predict(X_test)


accuracy_rfc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)


print('Accuracy of RFC Model - Depression: {:.5f}'.format(accuracy_rfc))
print('Confusion Matrix of RFC Model - Depression: \n', conf_matrix)
print('Classification report of RFC Model - Depression: \n', classif_report)
```

```
Accuracy of RFC Model - Depression: 0.97561
Confusion Matrix of RFC Model - Depression:
 [[116   0   0   0   0]
 [  0  12   3   1   0]
 [  0   1  37   0   0]
 [  0   0   2  74   0]
 [  0   0   0   0  41]]
Classification report of RFC Model - Depression:
                   precision    recall  f1-score   support

Extremely Severe       1.00      1.00      1.00       116
         Minimal       0.92      0.75      0.83        16
        Moderate       0.88      0.97      0.93        38
          Normal       0.99      0.97      0.98        76
          Severe       1.00      1.00      1.00        41

        accuracy                           0.98       287
       macro avg       0.96      0.94      0.95       287
    weighted avg       0.98      0.98      0.98       287
```

In [167…

```
accuracy_depression = [accuracy_knn, accuracy_svc, accuracy_nb, accuracy_rfc ]

depression_analysis = pd.DataFrame({
                        'Models' : ['KNN', 'SVC', 'Naive Bayes', 'Random Forest'],
                        'Accuracy_Depression': [x * 100 for x in accuracy_depressio
                      })
```

Data Classification Models for Anxiety

Data Pre-processing

In [168…

```
Y = df['CategoryAnx']
X = df.drop(columns=['Q2A','Q4A','Q7A','Q9A','Q15A','Q19A','Q20A','Q23A','Q25A','Q2
X.head()
```

Out[168…

	source	TIPI1	TIPI2	TIPI3	TIPI4	TIPI5	TIPI6	TIPI7	TIPI8	TIPI9	...	gender	age
26	2	4	1	7	7	7	2	6	5	1	...	2	15
31	2	5	5	5	5	7	7	7	5	5	...	1	18
57	0	2	4	1	5	6	4	3	6	6	...	2	21
81	2	7	6	3	6	4	1	7	5	3	...	2	14
87	2	1	3	5	6	6	7	2	7	1	...	1	17

5 rows × 23 columns

In [169…

```
scaler = StandardScaler()
scaler.fit(X)
```

```
X = scaler.transform(X)
```

In [170…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_st
```

Classification Models For Anxiety

In [171…
```
# KNN Classification

knn = KNeighborsClassifier(n_neighbors=95)


knn.fit(X_train, Y_train)


Y_pred = knn.predict(X_test)


accuracy_knn = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of KNN Model - Anxiety: {:.5f}'.format(accuracy_knn))
print('Confusion Matrix of KNN Model - Anxiety: \n', conf_matrix)
print('Confusion Matrix of KNN Model - Anxiety: \n', classif_report)
```

```
Accuracy of KNN Model - Anxiety: 0.63066
Confusion Matrix of KNN Model - Anxiety:
 [[112   0   0   0   0]
 [  9   0   0  11   0]
 [ 24   0   0  10   0]
 [ 17   0   0  69   0]
 [ 35   0   0   0   0]]
Confusion Matrix of KNN Model - Anxiety:
                   precision    recall  f1-score   support

Extremely Severe       0.57      1.00      0.72       112
         Minimal       0.00      0.00      0.00        20
        Moderate       0.00      0.00      0.00        34
          Normal       0.77      0.80      0.78        86
          Severe       0.00      0.00      0.00        35

        accuracy                           0.63       287
       macro avg       0.27      0.36      0.30       287
    weighted avg       0.45      0.63      0.52       287
```

In [172…
```
# SVC Model for Anxiety Classification
svm_model = SVC(kernel='linear')

svm_model.fit(X_train, Y_train)

preds = svm_model.predict(X_test)
```

```
accuracy_svc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of SVM Model - Anxiety: {:.5f}'.format(accuracy_svc))
print('Confusion Matrix of SVM Model - Anxiety: \n', conf_matrix)
print('Confusion Matrix of SVM Model - Anxiety: \n', classif_report)
```

```
Accuracy of SVM Model - Anxiety: 0.63066
Confusion Matrix of SVM Model - Anxiety:
 [[112   0   0   0   0]
 [  9   0   0  11   0]
 [ 24   0   0  10   0]
 [ 17   0   0  69   0]
 [ 35   0   0   0   0]]
Confusion Matrix of SVM Model - Anxiety:
                   precision    recall  f1-score   support

Extremely Severe       0.57      1.00      0.72       112
         Minimal       0.00      0.00      0.00        20
        Moderate       0.00      0.00      0.00        34
          Normal       0.77      0.80      0.78        86
          Severe       0.00      0.00      0.00        35

        accuracy                           0.63       287
       macro avg       0.27      0.36      0.30       287
    weighted avg       0.45      0.63      0.52       287
```

In [173…

```
# Naive Bayes Model for Anxiety Classification
clfNB = GaussianNB()


clfNB.fit(X_train, Y_train)


Y_pred = clfNB.predict(X_test)


accuracy_nb = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of NB Model - Anxiety: {:.5f}'.format(accuracy_nb))
print('Confusion Matrix of NB Model - Anxiety: \n', conf_matrix)
print('Classification Report of NB Model - Anxiety: \n', classif_report)
```

```
Accuracy of NB Model - Anxiety: 0.97213
Confusion Matrix of NB Model - Anxiety:
 [[110   0   0   0   2]
 [  1  19   0   0   0]
 [  0   0  32   0   2]
 [  1   0   0  85   0]
 [  2   0   0   0  33]]
Classification Report of NB Model - Anxiety:
                   precision    recall  f1-score   support

Extremely Severe       0.96      0.98      0.97       112
         Minimal       1.00      0.95      0.97        20
        Moderate       1.00      0.94      0.97        34
          Normal       1.00      0.99      0.99        86
          Severe       0.89      0.94      0.92        35

        accuracy                           0.97       287
       macro avg       0.97      0.96      0.97       287
    weighted avg       0.97      0.97      0.97       287
```

In [174…

```python
# Random Forest Classifier Model for Anxiety Classification
clfRFC = RandomForestClassifier(n_estimators=95, random_state=85)


clfRFC.fit(X_train, Y_train)


Y_pred = clfRFC.predict(X_test)


accuracy_rfc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)


print('Accuracy of RFC Model - Anxiety: {:.5f}'.format(accuracy_rfc))
print('Confusion Matrix of RFC Model - Anxiety: \n', conf_matrix)
print('Classification report of KNN Model - Anxiety: \n', classif_report)
```

```
Accuracy of RFC Model - Anxiety: 0.94077
Confusion Matrix of RFC Model - Anxiety:
 [[112   0   0   0   0]
 [  0   3  15   2   0]
 [  0   0  34   0   0]
 [  0   0   0  86   0]
 [  0   0   0   0  35]]
Classification report of KNN Model - Anxiety:
                   precision    recall  f1-score   support

Extremely Severe       1.00      1.00      1.00       112
         Minimal       1.00      0.15      0.26        20
        Moderate       0.69      1.00      0.82        34
          Normal       0.98      1.00      0.99        86
          Severe       1.00      1.00      1.00        35

        accuracy                           0.94       287
       macro avg       0.93      0.83      0.81       287
    weighted avg       0.96      0.94      0.92       287
```

In [175…
```
accuracy_anxiety = [accuracy_knn, accuracy_svc, accuracy_nb, accuracy_rfc ]

anxiety_analysis = pd.DataFrame({
                        'Models' : ['KNN', 'SVC', 'Naive Bayes', 'Random Forest' ],
                        'Accuracy_Anxiety': [x * 100 for x in accuracy_anxiety]
                    })
```

Data Classification Models for Stress

Data Preprocessing

In [176…
```
Y = df['CategoryStre']
X = df.drop(columns=['Q2A','Q4A','Q7A','Q9A','Q15A','Q19A','Q20A','Q23A','Q25A','Q2
X.head()
```

Out[176…

	source	TIPI1	TIPI2	TIPI3	TIPI4	TIPI5	TIPI6	TIPI7	TIPI8	TIPI9	...	gender	age
26	2	4	1	7	7	7	2	6	5	1	...	2	15
31	2	5	5	5	5	7	7	7	5	5	...	1	18
57	0	2	4	1	5	6	4	3	6	6	...	2	21
81	2	7	6	3	6	4	1	7	5	3	...	2	14
87	2	1	3	5	6	6	7	2	7	1	...	1	17

5 rows × 23 columns

In [177…
```
scaler = StandardScaler()
scaler.fit(X)
```

```
X = scaler.transform(X)
```

In [178…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_st
```

Classification Models For Stress

In [179…
```
# KNN Model for Stress Classification


knn = KNeighborsClassifier(n_neighbors=95)


knn.fit(X_train, Y_train)


Y_pred = knn.predict(X_test)


accuracy_knn = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of KNN Model - Stress: {:.5f}'.format(accuracy_knn))
print('Confusion Matrix of KNN Model - Stress: \n', conf_matrix)
print('Confusion Matrix of KNN Model - Stress: \n', classif_report)
```

```
Accuracy of KNN Model - Stress: 0.49477
Confusion Matrix of KNN Model - Stress:
 [[14  0  0  1 45]
 [ 0  0 10 13 12]
 [ 1  0 11 10 26]
 [ 0  0 11 72  3]
 [ 0  0  6  7 45]]
Confusion Matrix of KNN Model - Stress:
                   precision    recall  f1-score   support

Extremely Severe       0.93      0.23      0.37        60
         Minimal       0.00      0.00      0.00        35
        Moderate       0.29      0.23      0.26        48
          Normal       0.70      0.84      0.76        86
          Severe       0.34      0.78      0.48        58

        accuracy                           0.49       287
       macro avg       0.45      0.42      0.37       287
    weighted avg       0.52      0.49      0.45       287
```

In [180…
```
# SVC Model for Stress Classification
svm_model = SVC(kernel='linear')

svm_model.fit(X_train, Y_train)

preds = svm_model.predict(X_test)
```

```
accuracy_svc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of SVM Model - Anxiety: {:.5f}'.format(accuracy_svc))
print('Confusion Matrix of SVM Model - Anxiety: \n', conf_matrix)
print('Confusion Matrix of SVM Model - Anxiety: \n', classif_report)
```

```
Accuracy of SVM Model - Anxiety: 0.49477
Confusion Matrix of SVM Model - Anxiety:
 [[14  0  0  1 45]
 [ 0  0 10 13 12]
 [ 1  0 11 10 26]
 [ 0  0 11 72  3]
 [ 0  0  6  7 45]]
Confusion Matrix of SVM Model - Anxiety:
                   precision    recall  f1-score   support

Extremely Severe       0.93      0.23      0.37        60
         Minimal       0.00      0.00      0.00        35
        Moderate       0.29      0.23      0.26        48
          Normal       0.70      0.84      0.76        86
          Severe       0.34      0.78      0.48        58

        accuracy                           0.49       287
       macro avg       0.45      0.42      0.37       287
    weighted avg       0.52      0.49      0.45       287
```

In [181…

```
# Naive Bayes Model for Stress Classification

clfNB = GaussianNB()


clfNB.fit(X_train, Y_train)


Y_pred = clfNB.predict(X_test)


accuracy_nb = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)

print('Accuracy of NB Model - Stress: {:.5f}'.format(accuracy_nb))
print('Confusion Matrix of NB Model - Stress: \n', conf_matrix)
print('Classification Report of KNN Model - Stress: \n', classif_report)
```

```
Accuracy of NB Model - Stress: 0.93728
Confusion Matrix of NB Model - Stress:
 [[59  0  0  0  1]
 [ 0 32  0  3  0]
 [ 1  1 43  1  2]
 [ 0  1  0 85  0]
 [ 3  0  5  0 50]]
Classification Report of KNN Model - Stress:
                   precision    recall  f1-score   support

Extremely Severe       0.94      0.98      0.96        60
         Minimal       0.94      0.91      0.93        35
        Moderate       0.90      0.90      0.90        48
          Normal       0.96      0.99      0.97        86
          Severe       0.94      0.86      0.90        58

        accuracy                           0.94       287
       macro avg       0.93      0.93      0.93       287
    weighted avg       0.94      0.94      0.94       287
```

In [182…

```python
# Random Forest Classifier Model for Stress Classification

clfRFC = RandomForestClassifier(n_estimators=95, random_state=85)


clfRFC.fit(X_train, Y_train)


Y_pred = clfRFC.predict(X_test)


accuracy_rfc = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
classif_report = classification_report(Y_test, Y_pred)


print('Accuracy of RFC Model - Stress: {:.5f}'.format(accuracy_rfc))
print('Confusion Matrix of RFC Model - Stress: \n', conf_matrix)
print('Classification report of KNN Model - Stress: \n', classif_report)
```

```
Accuracy of RFC Model - Stress: 0.96516
Confusion Matrix of RFC Model - Stress:
 [[60  0  0  0  0]
 [ 0 25 10  0  0]
 [ 0  0 48  0  0]
 [ 0  0  0 86  0]
 [ 0  0  0  0 58]]
Classification report of KNN Model - Stress:
                   precision    recall  f1-score   support

Extremely Severe       1.00      1.00      1.00        60
         Minimal       1.00      0.71      0.83        35
        Moderate       0.83      1.00      0.91        48
          Normal       1.00      1.00      1.00        86
          Severe       1.00      1.00      1.00        58

        accuracy                           0.97       287
       macro avg       0.97      0.94      0.95       287
    weighted avg       0.97      0.97      0.96       287
```

In [183…
```
accuracy_stress = [accuracy_knn, accuracy_svc, accuracy_nb, accuracy_rfc ]

stress_analysis = pd.DataFrame({
                        'Models' : ['KNN', 'SVC', 'Naive Bayes', 'Random Forest' ],
                        'Accuracy_Stress': [x * 100 for x in accuracy_stress]
                      })
```

Analysis of the Models Used

In [184…
```
analysis = pd.concat([stress_analysis.set_index('Models'),
                      anxiety_analysis.set_index('Models'),
                      depression_analysis.set_index('Models')],
                     axis=1)
```

In [185…
```
analysis.head()
```

Out[185…

Models	Accuracy_Stress	Accuracy_Anxiety	Accuracy_Depression
KNN	49.477352	63.066202	59.930314
SVC	49.477352	63.066202	59.930314
Naive Bayes	93.728223	97.212544	94.425087
Random Forest	96.515679	94.076655	97.560976

In [186…
```
ax = analysis.plot(kind='barh')
plt.title("Accuracy of classification using different ML algorithms for DASS42")
plt.xlabel("Accuracy")
```

```
plt.ylabel("Models")
plt.show()
```