Data importation

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

imp_data = pd.read_excel('./Data/Data_grouped.xlsx')

# Dataframe creation + drop na
df = pd.DataFrame(imp_data)
df = df.dropna()
df.head()
```

Out[1]:

	Sample	PR1	PU2	SS3	NU4	PE5	PR6	NU7	PE8	SS9	...	PE11	NU12	PR13	SS14	PU15	PE16	NU
0	Clinical	3	3	2	3	2	3	4	2	3	...	2	3	3	4	3	3	
1	Clinical	3	4	2	2	2	4	1	2	2	...	3	4	4	4	3	4	
2	Clinical	2	4	4	4	2	3	3	3	4	...	2	3	3	4	3	3	
3	Clinical	1	3	2	3	2	2	2	2	1	...	1	2	2	2	1	4	
4	Clinical	3	3	2	2	2	2	4	2	1	...	2	2	3	2	2	2	

5 rows × 21 columns

In [2]:
```
print("Total N =",df.shape[0])
df['Sample'].value_counts()
```

Out[2]:
```
Total N = 18953
Non-clinical    18568
Clinical          385
Name: Sample, dtype: int64
```

In [3]:
```
df['Sample'] = df['Sample'].apply(lambda x: 0 if x == "Non-clinical" else 1)
```

In [4]:
```
df['Sample'].value_counts()
```

Out[4]:
```
0    18568
1      385
Name: Sample, dtype: int64
```

In [5]:
```
Neg_urgency = (df['NU4'] + df['NU7'] + df['NU12'] + df['NU17']) / 4
Pos_urgency = (df['PU2'] + df['PU10'] + df['PU15'] + df['PU20']) / 4
Sensation_seeking = (df['SS3'] + df['SS9'] + df['SS14'] + df['SS18']) / 4
Premeditation = (df['PR1'] + df['PR6'] + df['PR13'] + df['PR19']) / 4
Perseverance = (df['PE5'] + df['PE8'] + df['PE11'] + df['PE16'] ) / 4

df['neg_urgency'] = Neg_urgency
df['pos_urgency'] = Pos_urgency
df['sensation_seeking'] = Sensation_seeking
df['premeditation'] = Premeditation
df['perseverance'] = Perseverance

final_df = df[['neg_urgency', 'pos_urgency', 'sensation_seeking', 'premeditation', 'persevera
final_df.head()
```

Out[5]:

	neg_urgency	pos_urgency	sensation_seeking	premeditation	perseverance	Sample
0	3.25	2.75	3.00	3.00	2.25	1
1	2.00	2.75	3.00	3.50	2.75	1
2	3.00	2.50	4.00	2.75	2.50	1
3	2.25	1.75	2.00	1.75	2.25	1
4	2.50	2.00	1.75	2.75	2.00	1

METHOD 1 : RESAMPLING DATA WITH SMOTE BEFORE SPLITTING DATA

Resampling using SMOTE

In [6]:
```
#!pip install imbalanced-learn
from imblearn.over_sampling import SMOTE

X_M1 = final_df.drop(['Sample'], axis=1)
y_M1 = final_df['Sample']


sm_M1 = SMOTE(random_state=42)

X_res_M1, y_res_M1 = sm_M1.fit_resample(X_M1, y_M1)

print("Total added clinical profiles = ", y_res_M1.value_counts()[1] - df['Sample'].value_cou
print("total sample size = ", X_res_M1.shape[0])
y_res_M1.value_counts()
```

```
Total added clinical profiles =  18183
total sample size =  37136
```

Out[6]:
```
1    18568
0    18568
Name: Sample, dtype: int64
```

Data split

In [7]:
```
from sklearn.model_selection import train_test_split

X_train_M1, X_test_M1, y_train_M1, y_test_M1 = train_test_split(X_res_M1, y_res_M1, test_size
                                                                random_state=42, stratify=y_r

print("TRAIN SET size = ", y_train_M1.shape[0])
print(y_train_M1.value_counts())
print("\n TEST SET size = ", y_test_M1.shape[0])
print(y_test_M1.value_counts())
```

```
TRAIN SET size =  24881
1    12441
0    12440
Name: Sample, dtype: int64

 TEST SET size =  12255
0    6128
1    6127
Name: Sample, dtype: int64
```

Model fit and accuracy

In [8]:
```
from sklearn.ensemble import RandomForestClassifier

clf_M1 = RandomForestClassifier(random_state=42)

clf_M1.fit(X_train_M1, y_train_M1)
```

Out[8]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

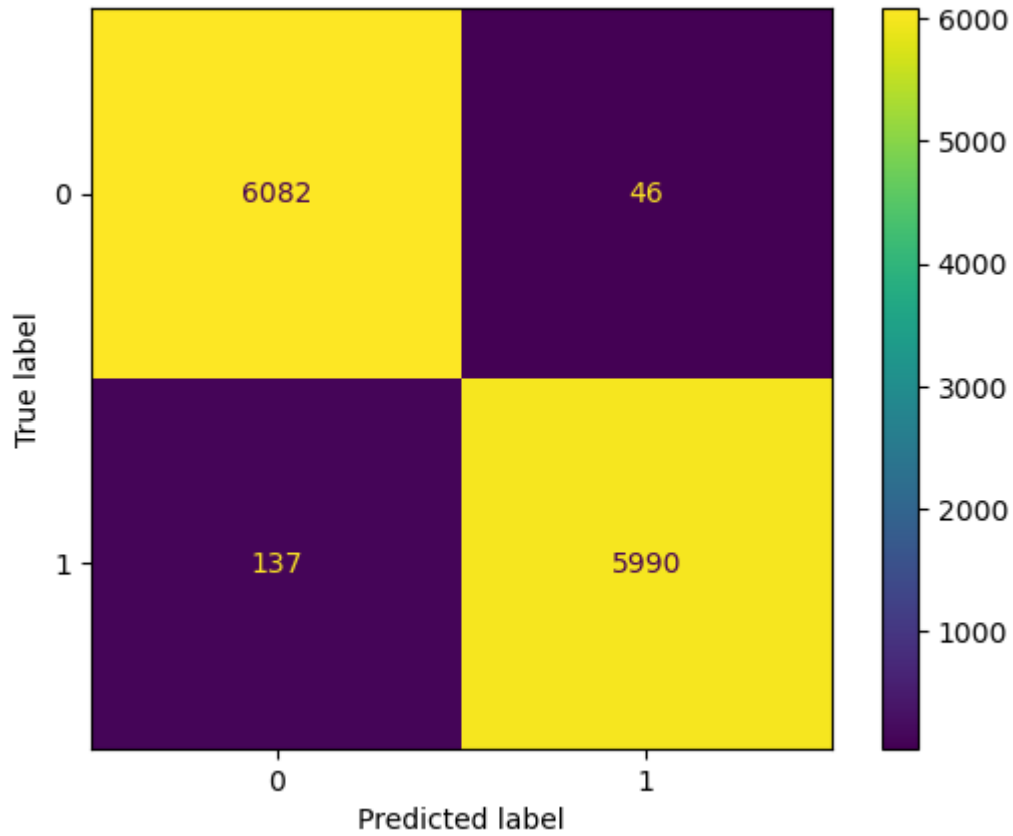
In [9]:
```
print('Accuracy = ', clf_M1.score(X_test_M1, y_test_M1).round(2))
```

```
Accuracy =  0.99
```

Confusion matrix and classification report

In [10]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_pred_M1 = clf_M1.predict(X_test_M1)

cm_M1 = confusion_matrix(y_test_M1, y_pred_M1, labels = clf_M1.classes_)

disp_M1 = ConfusionMatrixDisplay(confusion_matrix=cm_M1, display_labels=clf_M1.classes_)
disp_M1.plot()
```

Out[10]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1a525153fa0>
```

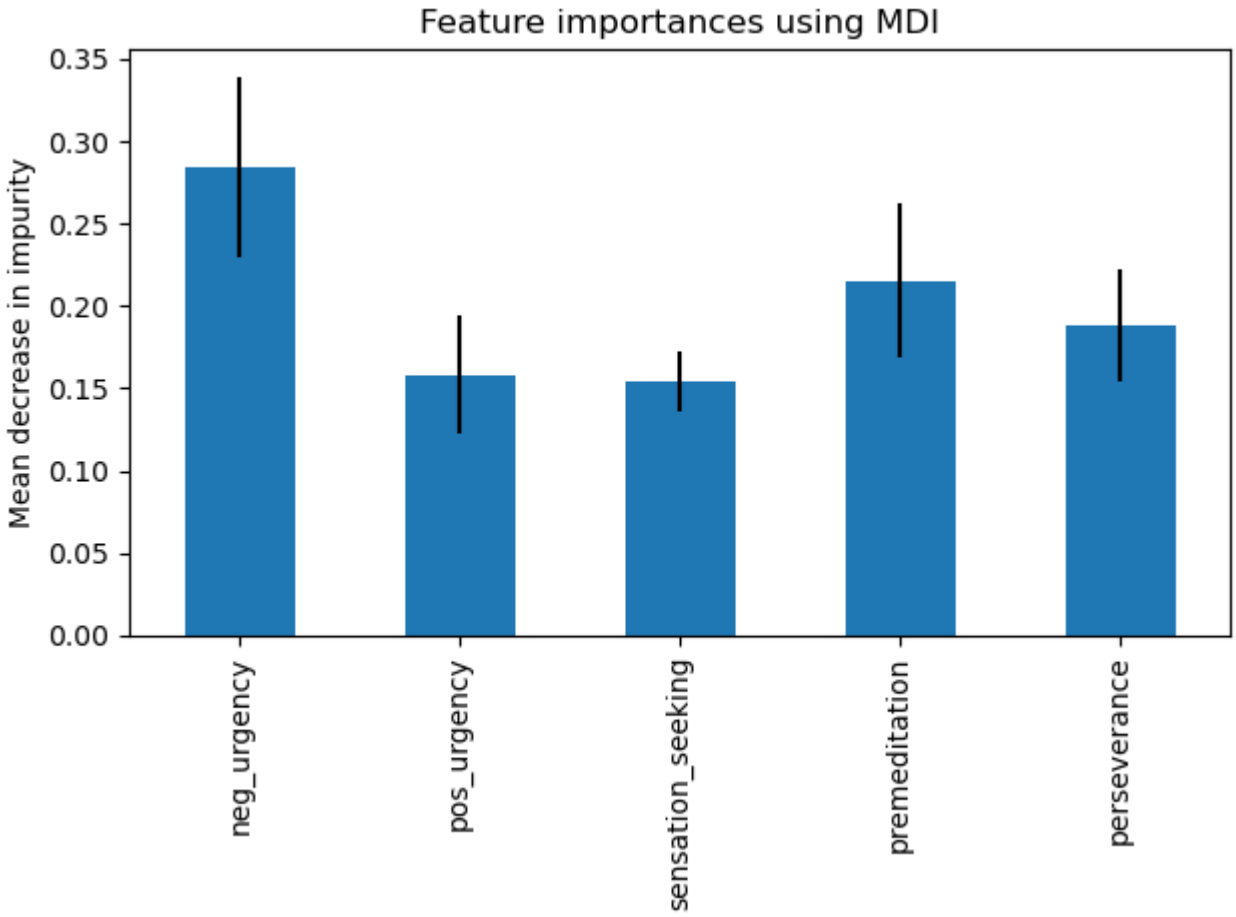


In [11]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test_M1, y_pred_M1))
```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	6128
1	0.99	0.98	0.98	6127
accuracy			0.99	12255
macro avg	0.99	0.99	0.99	12255
weighted avg	0.99	0.99	0.99	12255

Features importances

In [12]:
```
importances_M1 = clf_M1.feature_importances_
std_M1 = np.std([tree.feature_importances_ for tree in clf_M1.estimators_], axis=0)


forest_importances_M1 = pd.Series(importances_M1, index=X_M1.columns)

fig_M1, ax_M1 = plt.subplots()
forest_importances_M1.plot.bar(yerr=std_M1, ax=ax_M1)
ax_M1.set_title("Feature importances using MDI")
ax_M1.set_ylabel("Mean decrease in impurity")
fig_M1.tight_layout()
```



In [13]:
```
FI_report_M1_dic = {
    "Value": forest_importances_M1.round(3),
    "sd": std_M1.round(2)
}

pd.DataFrame.from_dict(FI_report_M1_dic)
```

Out[13]:

	Value	sd
neg_urgency	0.284	0.05
pos_urgency	0.158	0.04
sensation_seeking	0.154	0.02
premeditation	0.216	0.05
perseverance	0.188	0.03

METHOD 2 : RESAMPLING DATA WITH SMOTE AFTER SPLITTING DATA

Split the original dataset

In [14]:
```
from sklearn.model_selection import train_test_split

X_M2 = final_df.drop(['Sample'], axis=1)
y_M2 = final_df['Sample']

X_train_M2, X_test_M2, y_train_M2, y_test_M2 = train_test_split(X_M2, y_M2, test_size=0.33, r
print("TEST SAMPLE : \n", y_test_M2.value_counts())
print("Total size = ", y_test_M2.shape[0])
print("\n\n TRAIN SAMPLE : \n", y_train_M2.value_counts())
print("Total size = ", y_train_M2.shape[0])
```

```
TEST SAMPLE :
 0    6128
1     127
Name: Sample, dtype: int64
Total size =  6255


 TRAIN SAMPLE :
 0    12440
1      258
Name: Sample, dtype: int64
Total size =  12698
```

RESAMPLE WITH SMOTE ONLY THE TRAIN SET

In [15]:
```
from imblearn.over_sampling import SMOTE

sm_M2 = SMOTE(random_state=42)

X_res_M2, y_res_M2 = sm_M2.fit_resample(X_train_M2, y_train_M2)

print("Added profiles = ", y_res_M2.size - y_train_M2.size)
print("Total size after SMOTE = ", y_res_M2.shape[0])
y_res_M2.value_counts()
```

```
Added profiles =  12182
Total size after SMOTE =  24880
```

Out[15]:
```
0    12440
1    12440
Name: Sample, dtype: int64
```

Model fit and accuracy

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
```

```
clf_M2 = RandomForestClassifier(random_state=42)

clf_M2.fit(X_res_M2, y_res_M2)
```

Out[16]:

RandomForestClassifier

RandomForestClassifier(random_state=42)

In [17]:
```
print('Accuracy = ', clf_M2.score(X_test_M2, y_test_M2).round(2))
```

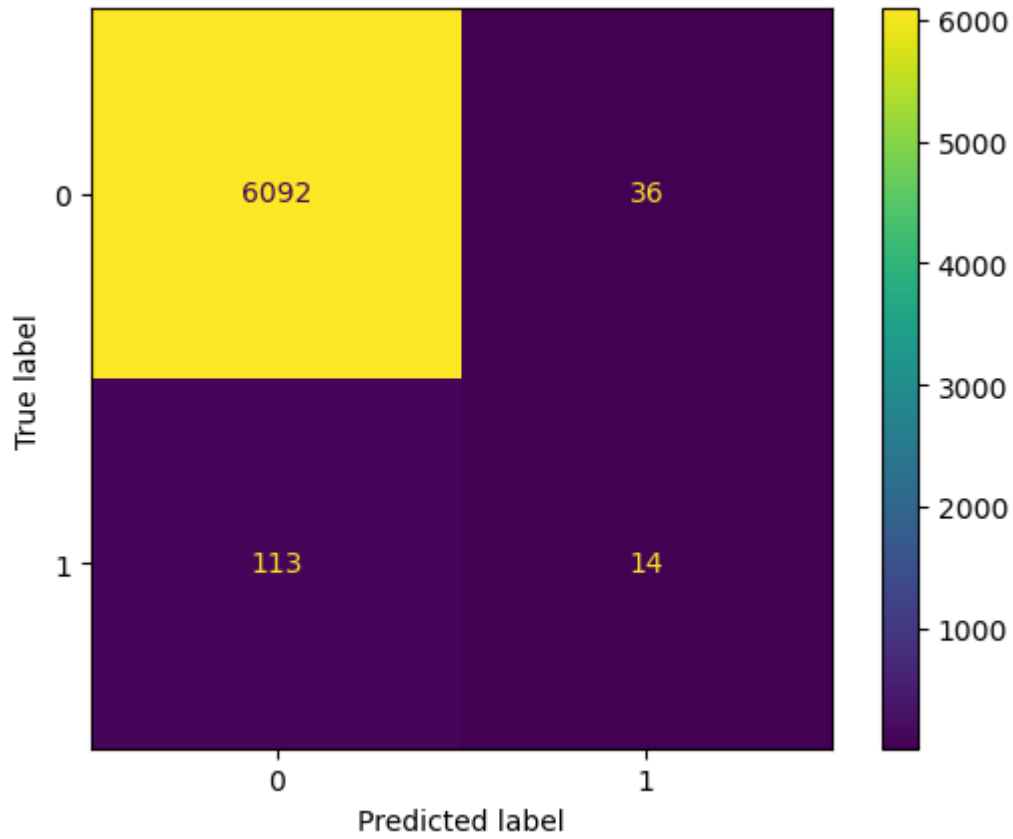
```
Accuracy =  0.98
```

Confusion matrix and classification report

In [18]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_pred_M2 = clf_M2.predict(X_test_M2)

cm_M2 = confusion_matrix(y_test_M2, y_pred_M2, labels = clf_M2.classes_)

disp_M2 = ConfusionMatrixDisplay(confusion_matrix=cm_M2, display_labels=clf_M2.classes_)
disp_M2.plot()
```

Out[18]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1a52bb05700>
```

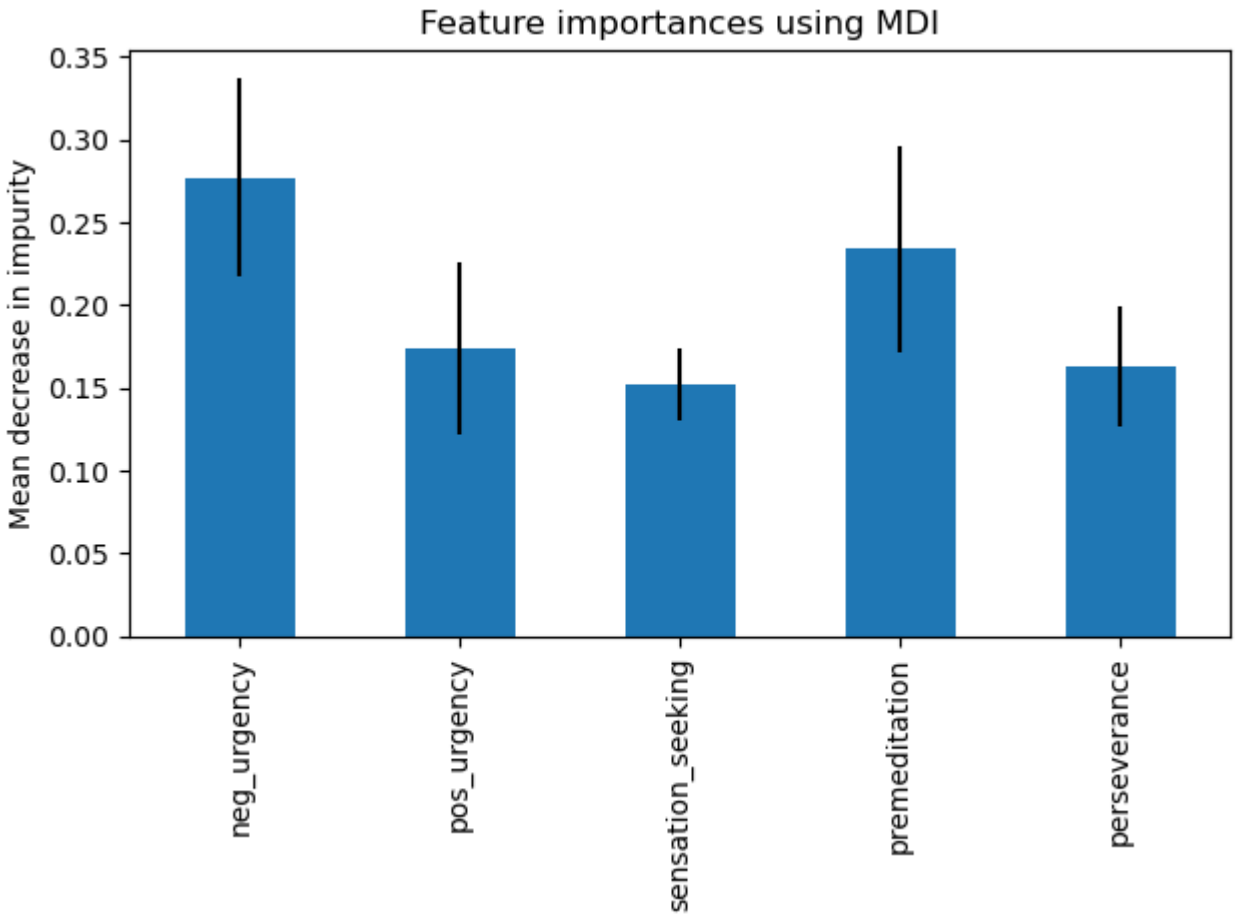


In [19]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test_M2, y_pred_M2))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      6128
           1       0.28      0.11      0.16       127

    accuracy                           0.98      6255
   macro avg       0.63      0.55      0.57      6255
weighted avg       0.97      0.98      0.97      6255
```

Features importances

In [20]:
```
importances_M2 = clf_M2.feature_importances_
std_M2 = np.std([tree.feature_importances_ for tree in clf_M2.estimators_], axis=0)


forest_importances_M2 = pd.Series(importances_M2, index=X_M2.columns)

fig_M2, ax_M2 = plt.subplots()
forest_importances_M2.plot.bar(yerr=std_M2, ax=ax_M2)
ax_M2.set_title("Feature importances using MDI")
ax_M2.set_ylabel("Mean decrease in impurity")
fig_M2.tight_layout()
```



In [21]:
```
FI_report_M2_dic = {
    "Value": forest_importances_M2.round(3),
    "sd": std_M2.round(2)
}

pd.DataFrame.from_dict(FI_report_M2_dic)
```

Out[21]:

	Value	sd
neg_urgency	0.277	0.06
pos_urgency	0.174	0.05
sensation_seeking	0.152	0.02
premeditation	0.234	0.06
perseverance	0.163	0.04