In [308…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, SelectPercentile, mutual_info_cl
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
```

In [309…

```
df=pd.read_csv('SaYoPillow.csv')
```

In [310…

```
df.head()
```

Out[310…

	snoring rate	respiration rate	body temperature	limb movement	blood oxygen	eye movement	sleeping hours	heart rate	stre lev
0	93.80	25.680	91.840	16.600	89.840	99.60	1.840	74.20	
1	91.64	25.104	91.552	15.880	89.552	98.88	1.552	72.76	
2	60.00	20.000	96.000	10.000	95.000	85.00	7.000	60.00	
3	85.76	23.536	90.768	13.920	88.768	96.92	0.768	68.84	
4	48.12	17.248	97.872	6.496	96.248	72.48	8.248	53.12	

Missing values

In [311…

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

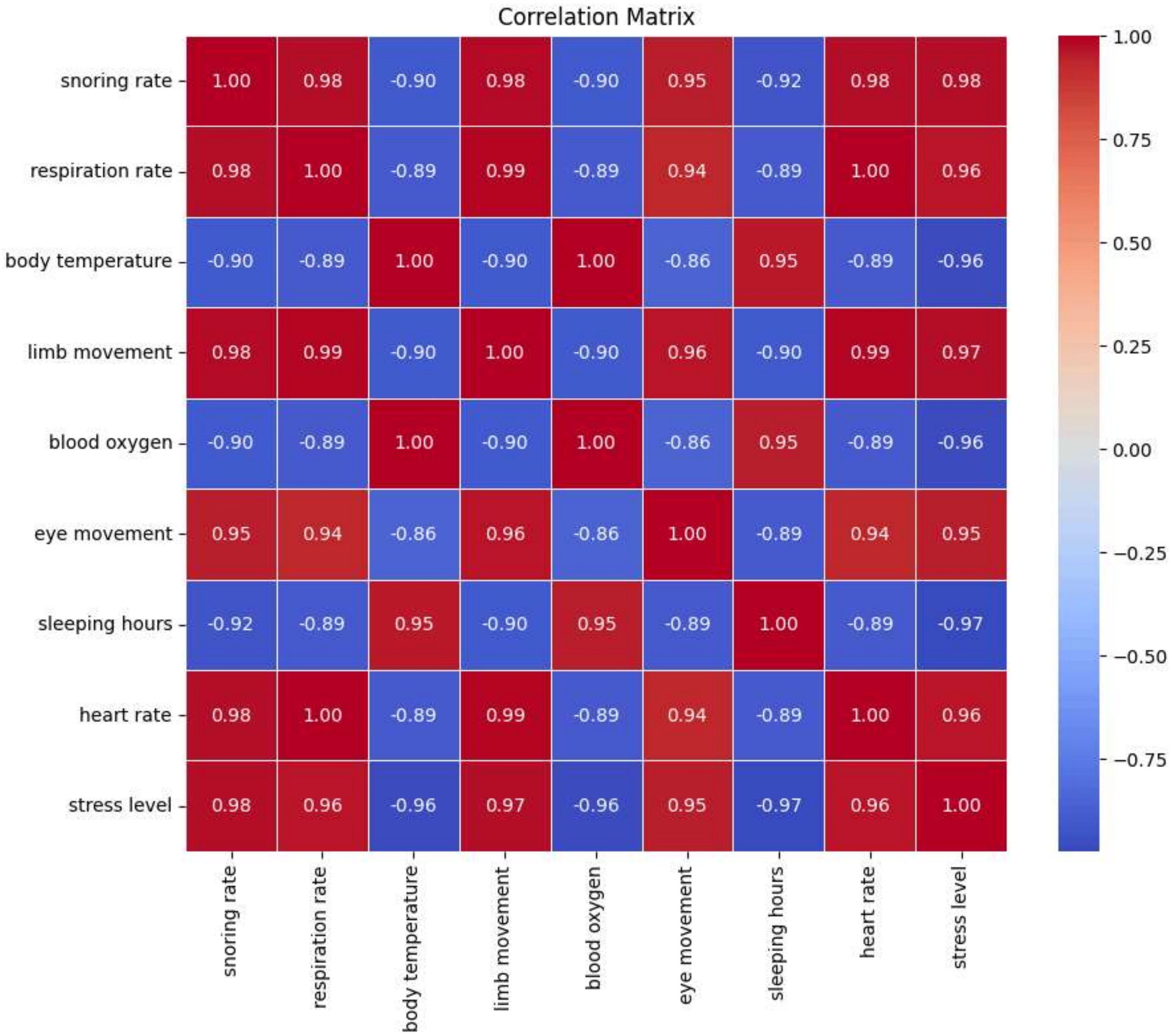
Out[311…

```
snoring rate         0
respiration rate     0
body temperature     0
limb movement        0
blood oxygen         0
eye movement         0
sleeping hours       0
heart rate           0
stress level         0
dtype: int64
```

Correlation Matrix

In [312…

```
corr = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```



	snoring rate	respiration rate	body temperature	limb movement	blood oxygen	eye movement	sleeping hours	heart rate	stress level
snoring rate	1.00	0.98	-0.90	0.98	-0.90	0.95	-0.92	0.98	0.98
respiration rate	0.98	1.00	-0.89	0.99	-0.89	0.94	-0.89	1.00	0.96
body temperature	-0.90	-0.89	1.00	-0.90	1.00	-0.86	0.95	-0.89	-0.96
limb movement	0.98	0.99	-0.90	1.00	-0.90	0.96	-0.90	0.99	0.97
blood oxygen	-0.90	-0.89	1.00	-0.90	1.00	-0.86	0.95	-0.89	-0.96
eye movement	0.95	0.94	-0.86	0.96	-0.86	1.00	-0.89	0.94	0.95
sleeping hours	-0.92	-0.89	0.95	-0.90	0.95	-0.89	1.00	-0.89	-0.97
heart rate	0.98	1.00	-0.89	0.99	-0.89	0.94	-0.89	1.00	0.96
stress level	0.98	0.96	-0.96	0.97	-0.96	0.95	-0.97	0.96	1.00

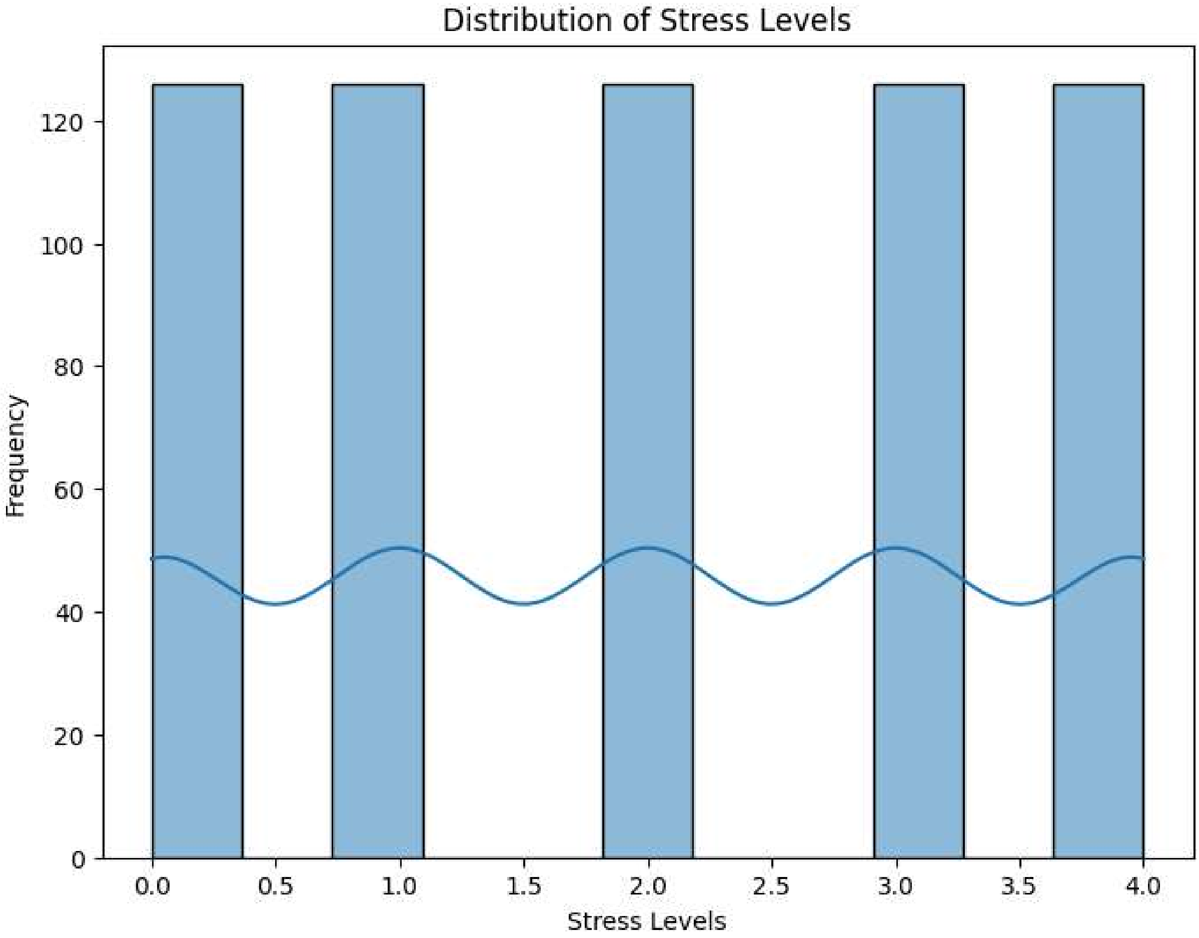
Stress Level Counts

In [313…

```
# Stress Level Counts
stress_level_counts = df['stress level'].value_counts()
print("Stress Level Counts:")
print(stress_level_counts)

# Visualize the distribution of Stress Levels
plt.figure(figsize=(8, 6))
sns.histplot(df['stress level'], kde=True)
plt.title('Distribution of Stress Levels')
plt.xlabel('Stress Levels')
plt.ylabel('Frequency')
plt.show()
```

```
Stress Level Counts:
stress level
3    126
1    126
0    126
2    126
4    126
Name: count, dtype: int64
```



In [314…

```
from sklearn.model_selection import train_test_split
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [315…

```
x.head()
```

Out[315…

	snoring rate	respiration rate	body temperature	limb movement	blood oxygen	eye movement	sleeping hours	heart rate
0	93.80	25.680	91.840	16.600	89.840	99.60	1.840	74.20
1	91.64	25.104	91.552	15.880	89.552	98.88	1.552	72.76
2	60.00	20.000	96.000	10.000	95.000	85.00	7.000	60.00
3	85.76	23.536	90.768	13.920	88.768	96.92	0.768	68.84
4	48.12	17.248	97.872	6.496	96.248	72.48	8.248	53.12

In [316…

```
xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=0.20)
```

Feature Selection 1

In [317…

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
```

In [318…

```
kbest = SelectKBest(mutual_info_classif, k=5)
select_feature = kbest.fit(xtrain ,ytrain)
```

In [319…

```
selected_features = xtrain.columns[select_feature.get_support()]
```

In [320…

```
xtrain.head()
```

Out[320…

	snoring rate	respiration rate	body temperature	limb movement	blood oxygen	eye movement	sleeping hours	heart rate
561	49.480	17.792	98.688	7.584	96.792	77.92	8.792	54.48
236	88.640	24.304	91.152	14.880	89.152	97.88	1.152	70.76
426	58.720	19.744	95.744	9.744	94.616	84.36	6.744	59.36
148	45.120	16.048	96.072	4.096	95.048	60.48	7.048	50.12
364	99.072	29.072	88.840	18.536	86.608	103.84	0.000	82.68

In [321…

```
x1=df[selected_features]
```

In [322…

```
x1train,x1test,y1train,y1test= train_test_split(x1,y,test_size=.20)
```

In [323…

```
model1=AdaBoostClassifier()
model1.fit(x1train,y1train)
```

Out[323…

```
▾ AdaBoostClassifier
AdaBoostClassifier()
```

In [324…

```
print('train score',model1.score(x1train,y1train))
print('test score',model1.score(x1test,y1test))
```

```
train score 0.998015873015873
test score 0.9603174603174603
```

In [352…

```
from sklearn.metrics import confusion_matrix

# Confusion matrix for Model 1
conf_matrix_model1 = confusion_matrix(ytest, model1.predict(x1test))
print("Confusion Matrix (Model 1):")
print(conf_matrix_model1)
```

```
Confusion Matrix (Model 1):
[[6 1 4 7 1]
 [5 8 3 6 4]
 [3 4 7 7 6]
 [9 5 6 3 7]
 [1 7 4 6 6]]
```

In [325…

```
from sklearn.metrics import classification_report

# Make predictions on the test set
y1pred = model1.predict(x1test)

# Generate classification report
print("Classification Report:")
print(classification_report(y1test, y1pred, zero_division=1))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        26
           1       0.92      1.00      0.96        23
           2       0.96      1.00      0.98        23
           3       0.93      0.96      0.95        28
           4       1.00      0.92      0.96        26

    accuracy                           0.96       126
   macro avg       0.96      0.96      0.96       126
weighted avg       0.96      0.96      0.96       126
```

Feature Selection 2

In [326…

```
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import mutual_info_classif
```

In [327…

```
sp = SelectPercentile(mutual_info_classif,percentile=50)
select_feature2 = sp.fit(xtrain,ytrain)
```

In [328…

```
selected_features2 = xtrain.columns[select_feature2.get_support()]
```

In [329…

```
x2 = df[selected_features2]
```

In [330…

```
x2train,x2test,y2train,y2test= train_test_split(x2,y,test_size=.20)
```

In [331…

```
model2=AdaBoostClassifier()
model2.fit(x2train,y2train)
```

Out[331…

```
▾ AdaBoostClassifier

AdaBoostClassifier()
```

In [332…

```
print('train score',model2.score(x2train,y2train))
print('test score',model2.score(x2test,y2test))
```

```
train score 0.6150793650793651
test score 0.5396825396825397
```

In [353…

```
# Confusion matrix for Model 2
conf_matrix_model2 = confusion_matrix(ytest, model2.predict(x2test))
print("\nConfusion Matrix (Model 2):")
print(conf_matrix_model2)
```

```
Confusion Matrix (Model 2):
[[ 4  0  0  8  7]
 [14  0  0  9  3]
 [15  0  0  9  3]
 [15  0  0 10  5]
 [12  0  0  8  4]]
```

In [333…

```
from sklearn.metrics import classification_report

# Make predictions on the test set
y2pred = model2.predict(x2test)

# Generate classification report
print("Classification Report:")
print(classification_report(y2test, y2pred, zero_division=1))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.43      1.00      0.60        26
           1       1.00      0.00      0.00        33
           2       1.00      0.00      0.00        25
           3       0.45      1.00      0.62        20
           4       1.00      1.00      1.00        22

    accuracy                           0.54       126
   macro avg       0.78      0.60      0.45       126
weighted avg       0.80      0.54      0.40       126
```

Feature importance

In [334…

```
feature_importance = model2.feature_importances_
feature_importance = pd.DataFrame(feature_importance, columns = ['Importance'])
```

In [335…

```
features = pd.DataFrame(xtrain.columns, columns = ['Feature'])
```

In [336…

```
importance_df = pd.concat([features, feature_importance],axis=1)
```

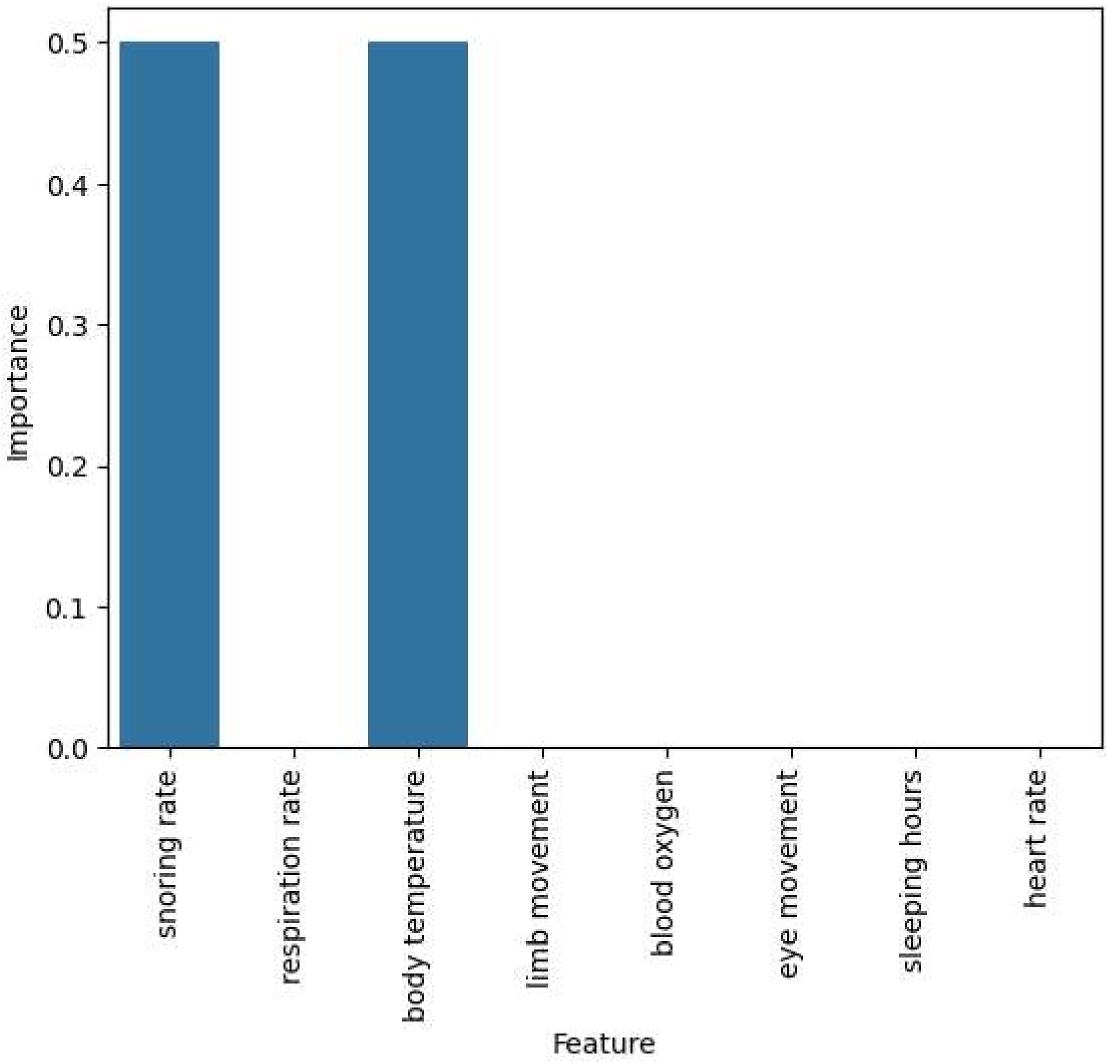
In [337…

```
print(importance_df)
```

```
              Feature  Importance
0        snoring rate         0.5
1    respiration rate         0.0
2    body temperature         0.5
3       limb movement         0.0
4        blood oxygen         NaN
5        eye movement         NaN
6      sleeping hours         NaN
7          heart rate         NaN
```

In [338…

```
sns.barplot(x=importance_df['Feature'], y=importance_df['Importance'])
plt.xticks(rotation=90)  # Rotate x-axis labels
plt.show()
```



In [339…

```
# Sort features by importance
importance_df_sorted = importance_df.sort_values(by='Importance', ascending=False)

# Select top N features based on importance
top_n_features = 2  # Specify the number of top features you want to select
selected_features = importance_df_sorted['Feature'][:top_n_features]
```

In [340…

```
# Create x3 DataFrame using the selected features
x3 = df[selected_features]
```

In [341…

```
x3.head()
```

Out[341…

	snoring rate	body temperature
0	93.80	91.840
1	91.64	91.552
2	60.00	96.000
3	85.76	90.768
4	48.12	97.872

In [342…

```
x3train,x3test,y3train,y3test = train_test_split(x3,y,test_size=.20)
```

In [343…

```
model3=AdaBoostClassifier()
model3.fit(x3train,y3train)
```

Out[343…

```
▾ AdaBoostClassifier
AdaBoostClassifier()
```

In [344…

```
print('trainscore',model3.score(x3train,y3train))
print('test score',model3.score(x3test,y3test))
```

```
trainscore 0.6091269841269841
test score 0.5634920634920635
```

In [354…

```
# Confusion matrix for Model 3
conf_matrix_model3 = confusion_matrix(y3test, model3.predict(x3test))
print("\nConfusion Matrix (Model 3):")
print(conf_matrix_model3)
```

```
Confusion Matrix (Model 3):
[[27  0  0  0  0]
 [ 1  0 26  0  0]
 [ 0  0 25  0  0]
 [ 0  0 28  0  0]
 [ 0  0  0  0 19]]
```

In [345…

```
from sklearn.metrics import classification_report

# Make predictions on the test set
y3pred = model3.predict(x3test)

# Generate classification report
print("Classification Report:")
print(classification_report(y3test, y3pred, zero_division=1))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        27
           1       1.00      0.00      0.00        27
           2       0.32      1.00      0.48        25
           3       1.00      0.00      0.00        28
           4       1.00      1.00      1.00        19

    accuracy                           0.56       126
   macro avg       0.86      0.60      0.49       126
weighted avg       0.86      0.56      0.46       126
```

SVM

In [346…

```python
from sklearn import svm
```

In [347…

```python
classifier=svm.SVC(kernel='linear',gamma='auto',C=3)
classifier.fit(xtrain,ytrain)
```

Out[347…

```
▾ SVC
SVC(C=3, gamma='auto', kernel='linear')
```

In [348…

```python
print('train score',classifier.score(xtrain,ytrain))
print('test score',classifier.score(xtest,ytest))
```

```
train score 1.0
test score 1.0
```

In [355…

```python
# Confusion matrix for SVM
conf_matrix_svm = confusion_matrix(ytest, classifier.predict(xtest))
print("\nConfusion Matrix (SVM):")
print(conf_matrix_svm)
```

```
Confusion Matrix (SVM):
[[19  0  0  0  0]
 [ 0 26  0  0  0]
 [ 0  0 27  0  0]
 [ 0  0  0 30  0]
 [ 0  0  0  0 24]]
```

In [349…

```python
from sklearn.metrics import classification_report

# Make predictions on the test set
ypred = classifier.predict(xtest)

# Generate classification report
print("Classification Report:")
print(classification_report(ytest, ypred, zero_division=1))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        26
           2       1.00      1.00      1.00        27
           3       1.00      1.00      1.00        30
           4       1.00      1.00      1.00        24

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Prediction

In [350…

```
# Example with values for each feature
manual_example = np.array([[20, 18, 37, 2, 95, 1, 7, 85]])  # Assuming 8 features a

# Display the manual example
print("manual_example:")
print(manual_example)
```

```
manual_example:
[[20 18 37  2 95  1  7 85]]
```

In [351…

```
# Prediction using Feature Selection 1
selected_features_model1 = x1.columns  # Features used in model1
manual_example_model1_df = pd.DataFrame(manual_example, columns=x.columns)  # Conve
manual_example_model1 = manual_example_model1_df[selected_features_model1]  # Selec
prediction_model1 = model1.predict(manual_example_model1)
print("Prediction (Feature Selection 1):", prediction_model1)

# Prediction using Feature Selection 2
selected_features_model2 = x2.columns  # Features used in model2
manual_example_model2_df = pd.DataFrame(manual_example, columns=x.columns)  # Conve
manual_example_model2 = manual_example_model2_df[selected_features_model2]  # Selec
prediction_model2 = model2.predict(manual_example_model2)
print("Prediction (Feature Selection 2):", prediction_model2)

# Prediction using Feature Importance (Model 3)
selected_features_model3 = x3.columns  # Features used in model3
manual_example_model3_df = pd.DataFrame(manual_example, columns=x.columns)  # Conve
manual_example_model3 = manual_example_model3_df[selected_features_model3]  # Selec
prediction_model3 = model3.predict(manual_example_model3)
print("Prediction (Feature Importance):", prediction_model3)

# Prediction using SVM
prediction_svm = classifier.predict(manual_example_df)  # No need to modify example
print("Prediction (SVM):", prediction_svm)
```

```
Prediction (Feature Selection 1): [2]
Prediction (Feature Selection 2): [0]
Prediction (Feature Importance): [0]
Prediction (SVM): [0]
```