Analyzing the Impact of Lifestyle Factors on Sleep Quality and Health

➢ Manikanta Pudoka
➢ Varshitha Mullangi
➢ Joshitha Dandlamudi
➢ Tejaswini Pallepati
➢ Fazaluddin Mohd

TABLE OF CONTENTS(HTML Link):

1.DATA DESCRIPTION

2.DATA ATTRIBUTES

3.SCATTER PLOT OF ALL FEATURES

4.SELECTED FEATURES

5.PLOTS

6.EDA

7.CONFUSION MATRIX

8.COEFFICIENTS OF LINEAR REG

9.KNN & NB

DATA DESCRIPTION:

This project utilizes the Sleep Health and Lifestyle Dataset from Kaggle, which contains information about individuals' sleep patterns, lifestyle choices, and health indicators.

Source: Kaggle (https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset)

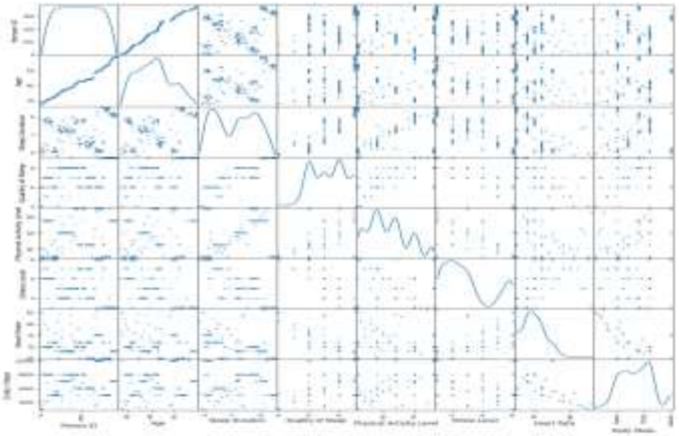
The dataset comprises records of individuals' sleep health and lifestyle factors. It includes various sleep-related metrics, daily habits, and health indicators that can potentially influence sleep quality and overall well-being.

DATA ATTRIBUTES:

1. BMI:This column contains the body mass index (BMI) category of the individuals, classified as:Normal,overweight and Obese.
2. Gender: It contains the gender of the individuals
3. Quality of sleep:A Subjective rating of sleep quality(Scale:1-10)
4. Heart Rate:It represents the number of heartbeats per minute (BPM)
5. Physical activity level:A rating of physical activity (Scale:1-100)
6. sleep disorder:mentioning the disorder types

SCATTER PLOT OF ALL FEATURES:

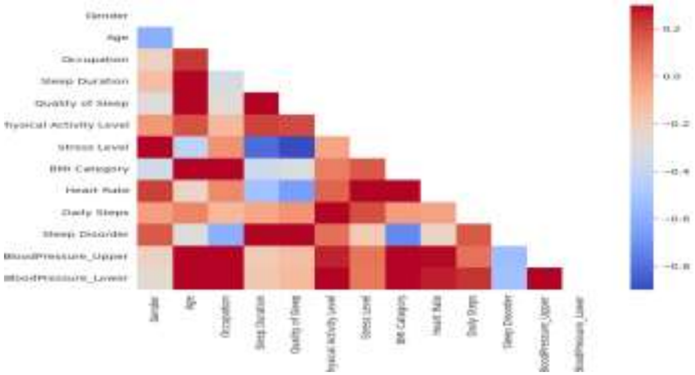
- Strong Positive Correlations: Physical Activity Level & Daily Steps, Quality of Sleep & Sleep Duration.
- Moderate Negative Correlations: Stress Level with Quality of Sleep & Physical Activity.
- Weak/No Correlations: Age and Person ID with most other variables.
- Key Observations: More physical activity increases steps; better sleep quality is linked to longer sleep; higher stress reduces sleep quality and physical activity.
- Potential Insights: Promoting physical activity and stress reduction may improve health metrics; sleep quality and duration are closely linked.



SELECTED FEATURES:

Positive Correlations:

- **Sleep Duration** and **Quality of Sleep:** These two features appear to be positively correlated. This suggests that longer sleep duration tends to be associated with better sleep quality.
- **Daily Steps** and **Physical Activity Level:** There is a strong positive correlation, which makes sense, as a higher number of daily steps likely reflects higher physical activity levels.
- **BloodPressure_Upper** and **BloodPressure_Lower:** These two measures of blood pressure are also highly positively correlated, as expected.

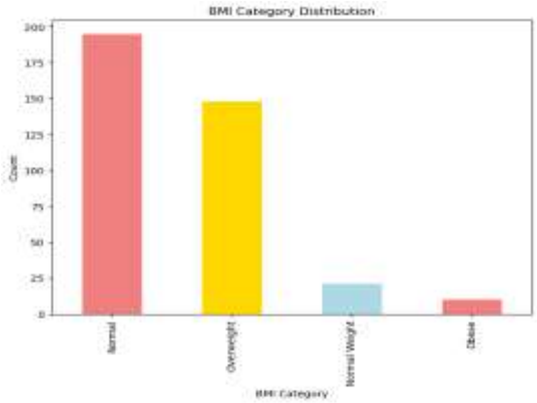


Negative Correlations:

- **Stress Level** and **Quality of Sleep**: These two variables show a strong negative correlation, indicating that higher stress levels are associated with poorer sleep quality.
- **Stress Level** and **Sleep Duration**: There's a negative relationship between these variables, suggesting that higher stress might reduce sleep duration.
- **Feature Selection Strategy:**
- **Variables to be selected :**
  - **Sleep Duration** (positive correlation with Quality of Sleep)
  - **Stress Level** (negative correlation with Quality of Sleep)
- Understanding how sleep duration and quality interact with stress and other factors can be critical in health, productivity, and lifestyle analysis.Along with the demographics features these can be selected.

BAR CHART:

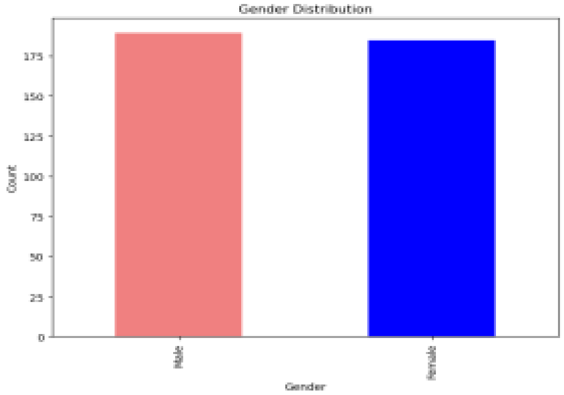
BMI CATEGORY: The population has a healthy weight distribution, with the majority in the Normal BMI category (195 individuals), followed by Overweight (150). The Normal Weight and Obese categories are smaller, with 25 and 10 individuals, respectively. There may be a need for weight management interventions for the Overweight group.



GENDER: The distribution between males and females is very close to equal.

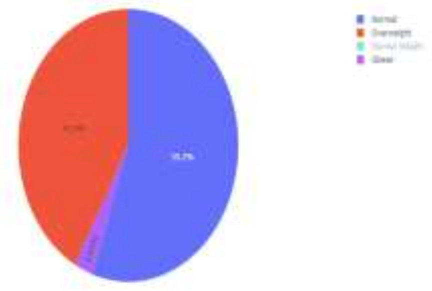
There is a slightly higher number of males than females, but the difference is minimal.

The y-axis (Count) starts at 0 and goes up to about 200, giving a clear representation of the actual numbers.



PIE CHART:

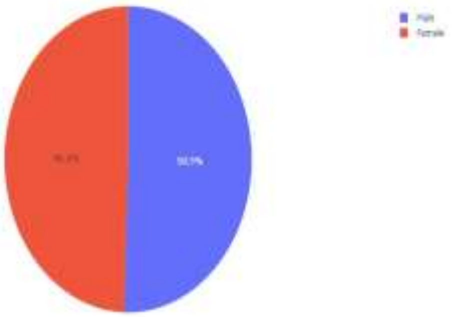
- BMI CATEGORY:Over half the population falls in the Normal BMI category
- Nearly 42% are classified as Overweight
- Combined Normal and Normal Weight categories make up 57.5%
- Obesity rate is very low at 0.6%



GENDER:

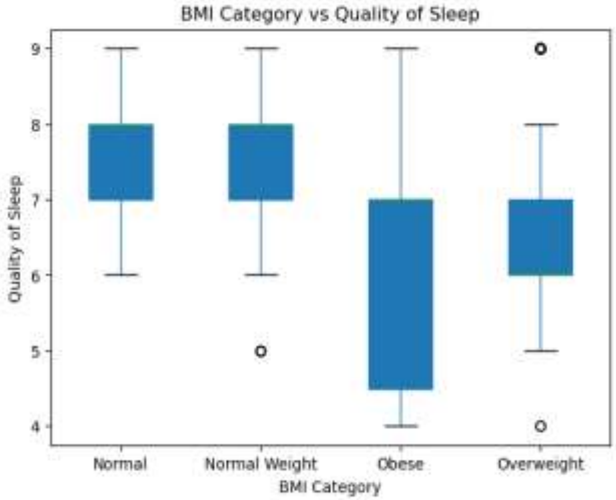
Nearly equal distribution between males and females

Slight male majority, but difference is minimal (1%)

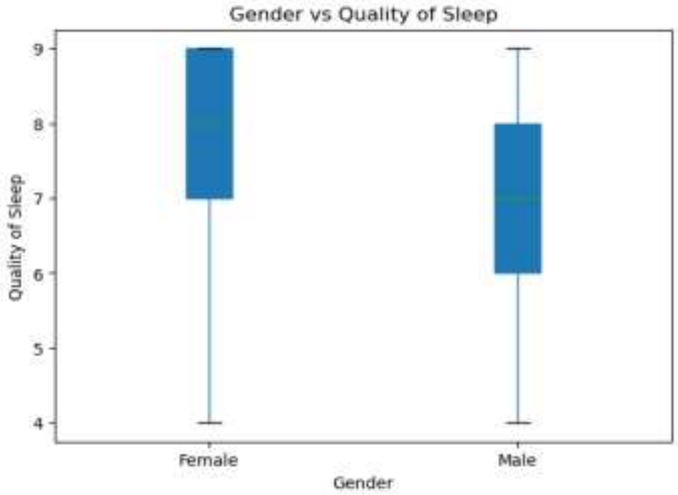


BOX PLOT:

- ➢ **Median:** Normal and Normal Weight categories have the highest sleep quality (7.5-8), while Obese has the lowest (5.5-6).
- ➢ **Spread:** Obese category shows the largest variability in sleep quality, followed by Overweight.
- ➢ **Range:** Obese category spans the widest range (4-9), while others range from 6 to 9.
- ➢ **Outliers:** Outliers are present in the Normal Weight and Overweight categories.
- ➢ **Overall Trend:** Sleep quality decreases as BMI increases, with more variability in higher BMI categories.
- ➢ **Skewness:** Overweight is slightly skewed towards lower sleep quality, while Obese shows slight skew towards higher values.

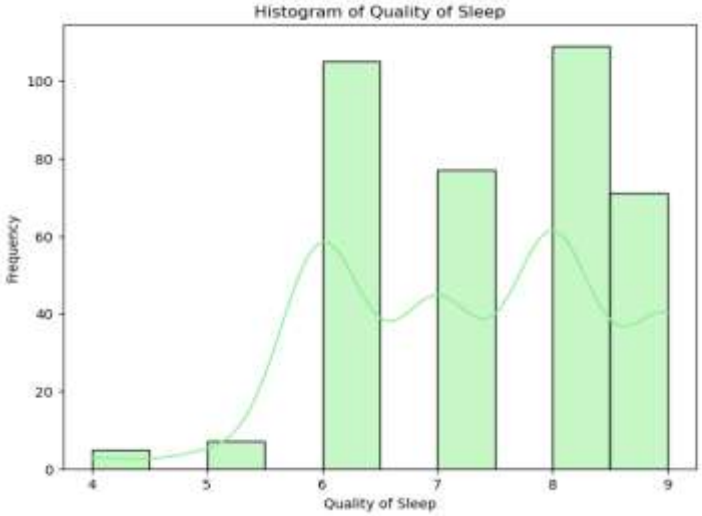


- **Median:** Females have a higher median sleep quality (8) than males (7).
- **Spread:** Female sleep quality is more consistent with a smaller interquartile range.
- **Range:** Both genders have similar overall ranges, from 4 to 9.
- **Skewness:** Female data is skewed towards lower values, while male data is more symmetrical.
- **Outliers:** No visible outliers for either gender.
- **Overall Comparison:** Females report slightly higher and more consistent sleep quality, while males show more variability

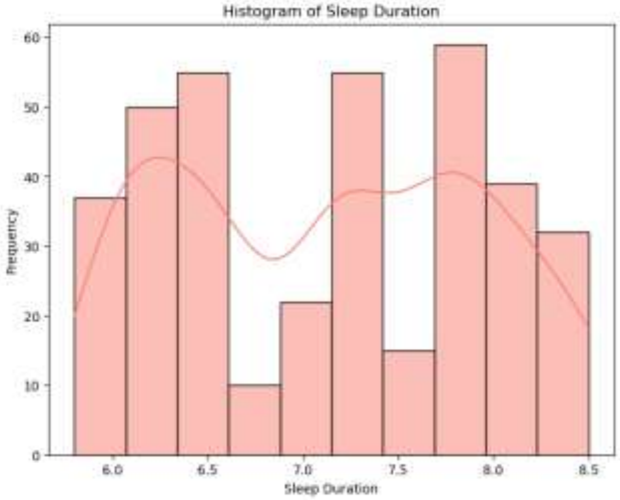


HISTOGRAM:

- ➢ **Range:** Sleep quality is rated from 4 to 9.
- ➢ **Distribution:** Multimodal with two peaks at 6 and 8, slightly right-skewed.
- ➢ **Peaks:** Most common ratings are 6 and 8, followed by 7.
- ➢ **Frequency:** Highest frequencies at 100-110 for 6 and 8 ratings.
- ➢ **Low/High Ratings:** Few report poor sleep (4-5), while many report good sleep (8-9).
- ➢ **Overall:** Most people experience average to good sleep quality, with no clear outliers.

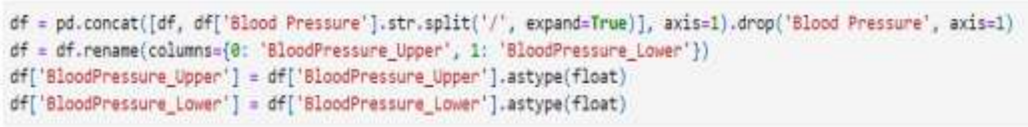


Most individuals sleep between 6.5 and 8.0 hours, with many meeting the recommended 7-9 hours. Some have shorter (6.0-6.5) or longer (8.0-8.5) sleep durations, indicating variability in sleep patterns. There's a noticeable dip in frequency around 7.0 hours. While most people fall within the healthy sleep range, subgroups may have sleep deficiencies or excess.

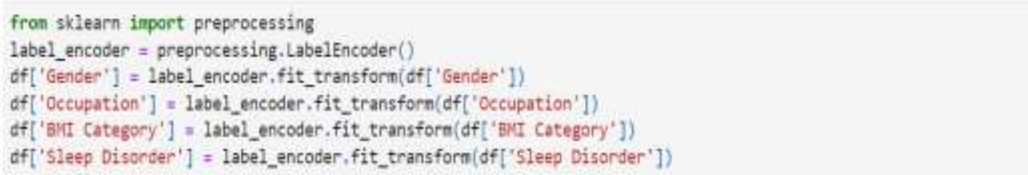


EDA

Data Preprocessing:



```
df = pd.concat([df, df['Blood Pressure'].str.split('/', expand=True)], axis=1).drop('Blood Pressure', axis=1)
df = df.rename(columns={0: 'BloodPressure_Upper', 1: 'BloodPressure_Lower'})
df['BloodPressure_Upper'] = df['BloodPressure_Upper'].astype(float)
df['BloodPressure_Lower'] = df['BloodPressure_Lower'].astype(float)
```

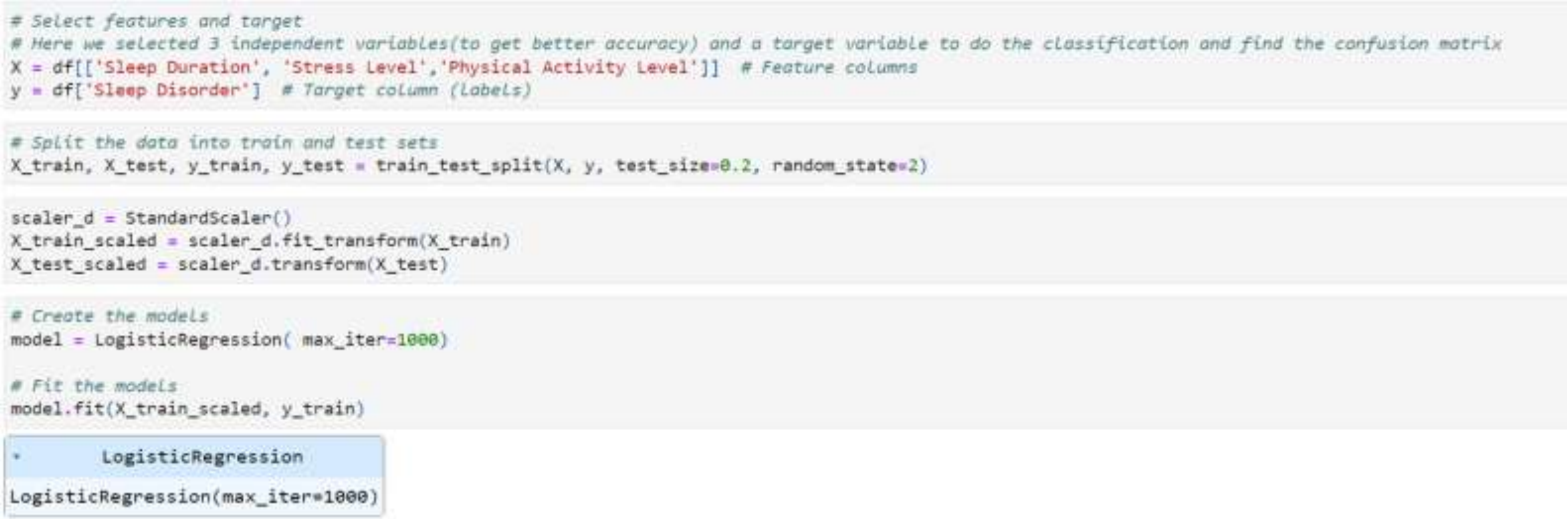


```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Occupation'] = label_encoder.fit_transform(df['Occupation'])
df['BMI Category'] = label_encoder.fit_transform(df['BMI Category'])
df['Sleep Disorder'] = label_encoder.fit_transform(df['Sleep Disorder'])
```

*The 'Blood Pressure' column is split into two new columns using string manipulation and are converted into the float type.

*Categorical features are converted into numeric values using Label encoder

Logistic Regression



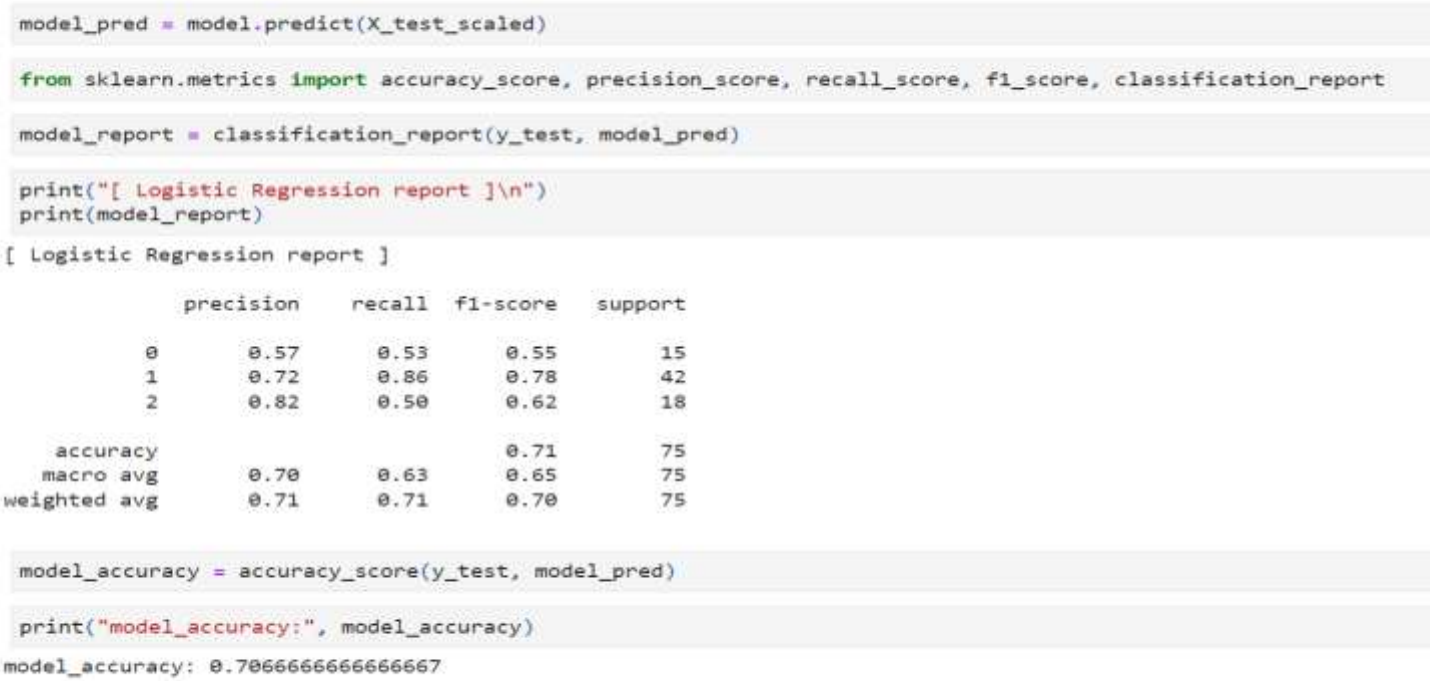
```
# Select features and target
# Here we selected 3 independent variables(to get better accuracy) and a target variable to do the classification and find the confusion matrix
X = df[['Sleep Duration', 'Stress Level','Physical Activity Level']]  # Feature columns
y = df['Sleep Disorder']  # Target column (labels)
```

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

```
scaler_d = StandardScaler()
X_train_scaled = scaler_d.fit_transform(X_train)
X_test_scaled = scaler_d.transform(X_test)
```

```
# Create the models
model = LogisticRegression( max_iter=1000)

# Fit the models
model.fit(X_train_scaled, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```



```python
model_pred = model.predict(X_test_scaled)
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
```

```python
model_report = classification_report(y_test, model_pred)
```

```python
print("[ Logistic Regression report ]\n")
print(model_report)
```

```
[ Logistic Regression report ]

              precision    recall  f1-score   support

           0       0.57      0.53      0.55        15
           1       0.72      0.86      0.78        42
           2       0.82      0.50      0.62        18

    accuracy                           0.71        75
   macro avg       0.70      0.63      0.65        75
weighted avg       0.71      0.71      0.70        75
```

```python
model_accuracy = accuracy_score(y_test, model_pred)
```

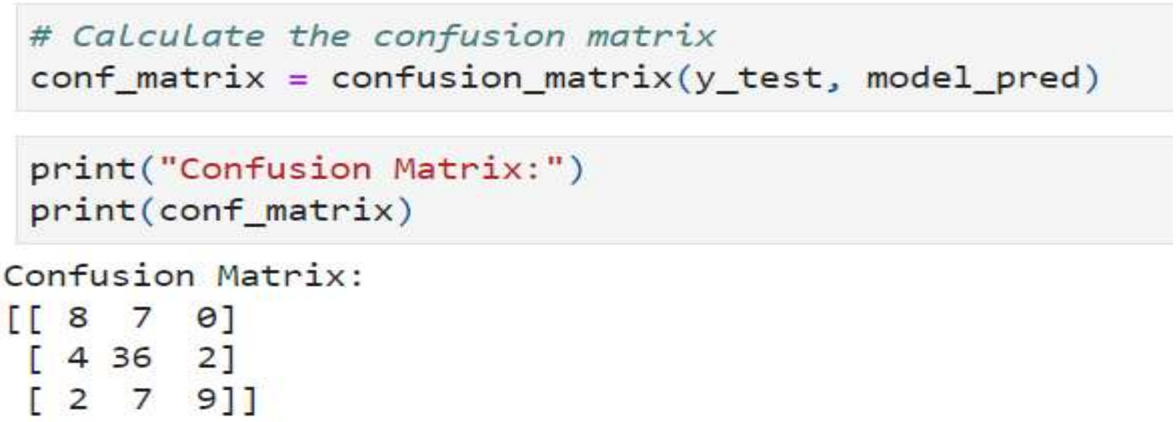
```python
print("model_accuracy:", model_accuracy)
```

```
model_accuracy: 0.7066666666666667
```

Results

1. **Overall Accuracy**: 71%
2. **Class Precision/Recall (Sleep Apnea, Insomnia, None)**:
   - ➢ Sleep Apnea (Class 1) has high recall, indicating it correctly identifies most cases of Sleep Apnea.
   - ➢ Insomnia (Class 2) has decent precision, but lower recall, meaning some cases of Insomnia are misclassified.
   - ➢ None (Class 0) has lower precision, with more misclassifications into other classes.

Confusion Matrix



```
# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test, model_pred)
```

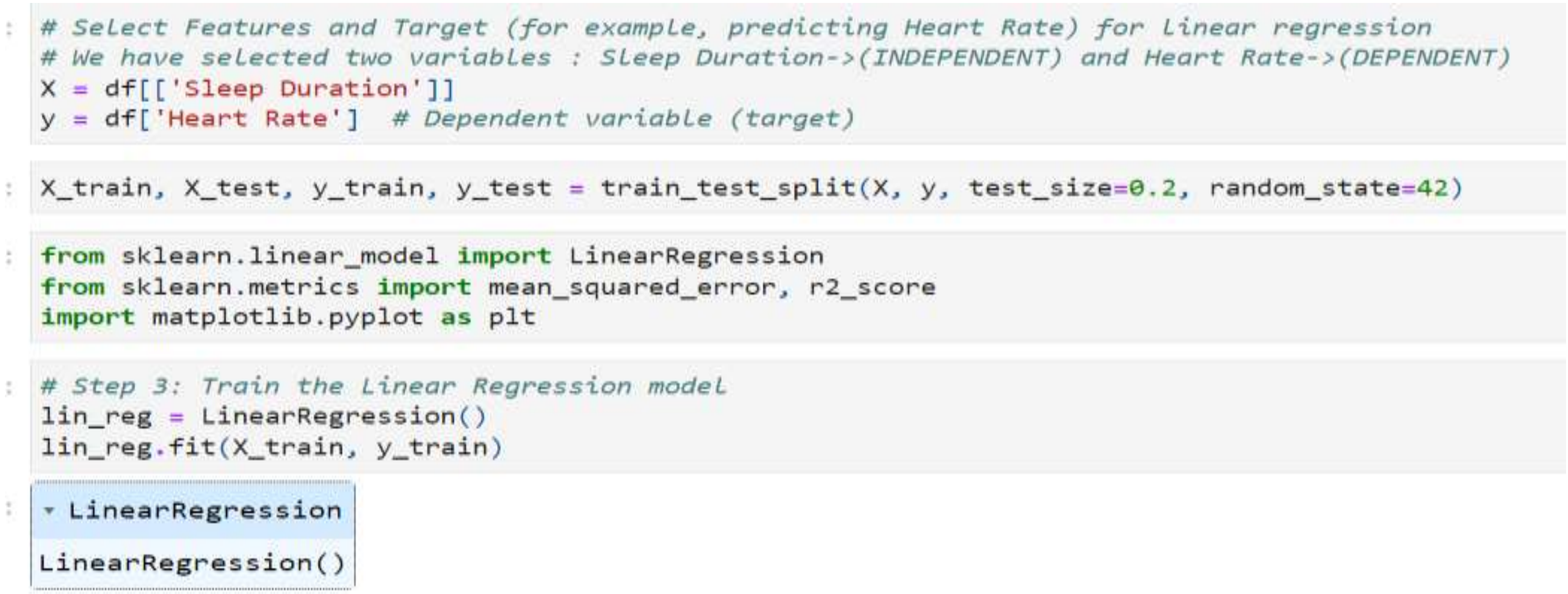
```
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[ 8  7  0]
 [ 4 36  2]
 [ 2  7  9]]
```

Interpretation

- Class 0 (None): The model is struggling with classifying "None" correctly, as there are 7 false negatives where "None" was predicted as "Sleep Apnea."
- Class 1 (Sleep Apnea): The model performs well at identifying Sleep Apnea, with 36 correct predictions and relatively fewer misclassifications compared to the other classes.
- Class 2 (Insomnia): The model has some trouble distinguishing between "Insomnia" and "Sleep Apnea", as 7 cases of "Insomnia" were incorrectly predicted as "Sleep Apnea."
- There are significant false negatives for the "None" class, leading to misclassification as "Sleep Apnea."
- "Insomnia" is frequently confused with "Sleep Apnea," indicating a challenge in differentiation between these two classes by the model.

Linear Regression



```
# Select Features and Target (for example, predicting Heart Rate) for Linear regression
# We have selected two variables : Sleep Duration->(INDEPENDENT) and Heart Rate->(DEPENDENT)
X = df[['Sleep Duration']]
y = df['Heart Rate']  # Dependent variable (target)
```

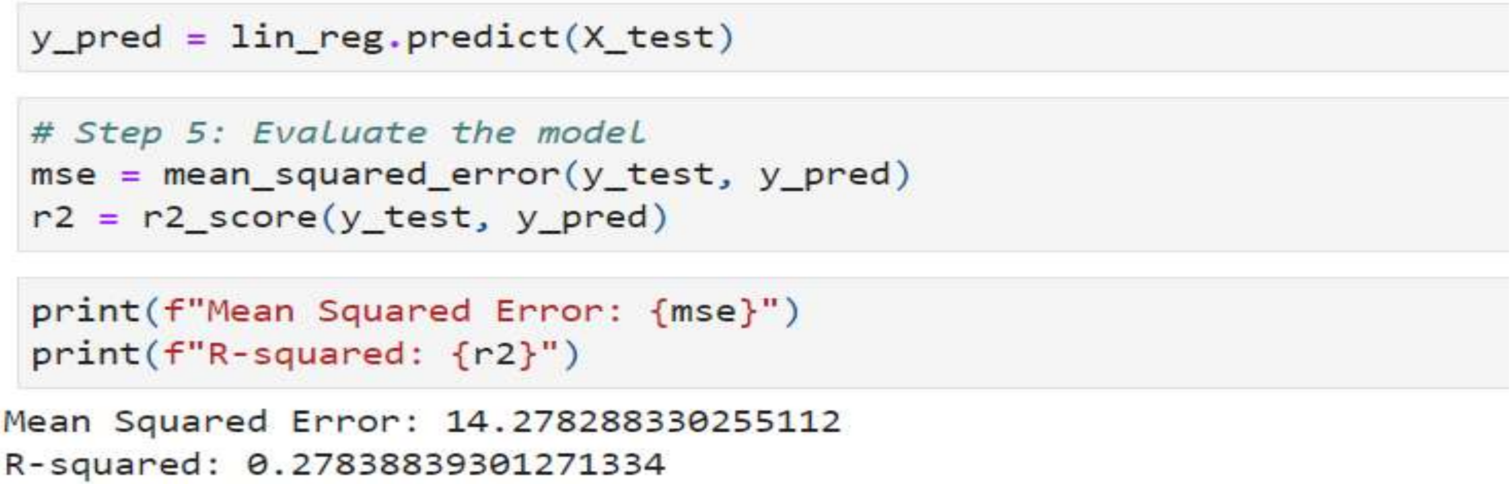
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

```
# Step 3: Train the Linear Regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

MSE and R-Squared



```python
y_pred = lin_reg.predict(X_test)
```

```python
# Step 5: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

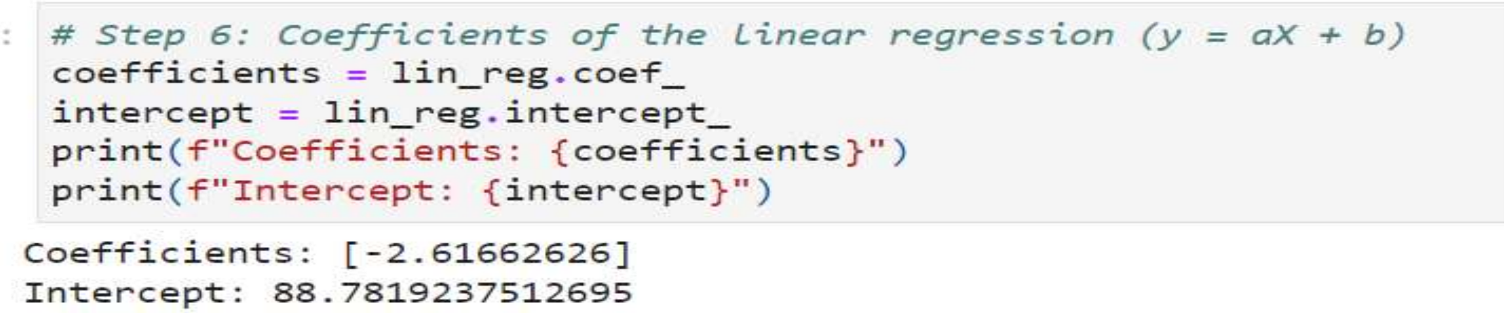
```python
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 14.278288330255112
R-squared: 0.27838839301271334
```

Interpretation

- The MSE value of **14.28** means that, on average, the squared difference between your predicted sleep outcomes (e.g., sleep disorder, sleep quality) and the actual outcomes is about 14.28 units.
- If you're predicting something like sleep quality, this means the predictions deviate from actual values by a substantial margin, leading to prediction errors.
- The R^2 value of **0.278** means that only **27.8%** of the variability in the target variable (such as sleep disorder or quality) is explained by the features in the model (e.g., sleep duration, stress level, physical activity).
- This low R^2 indicates that your model is not capturing most of the factors that contribute to variability in the sleep outcome. The remaining 72.2% of variability is unexplained, implying that other factors, which are not included in the model, might play a significant role.

Coefficients of Linear regression



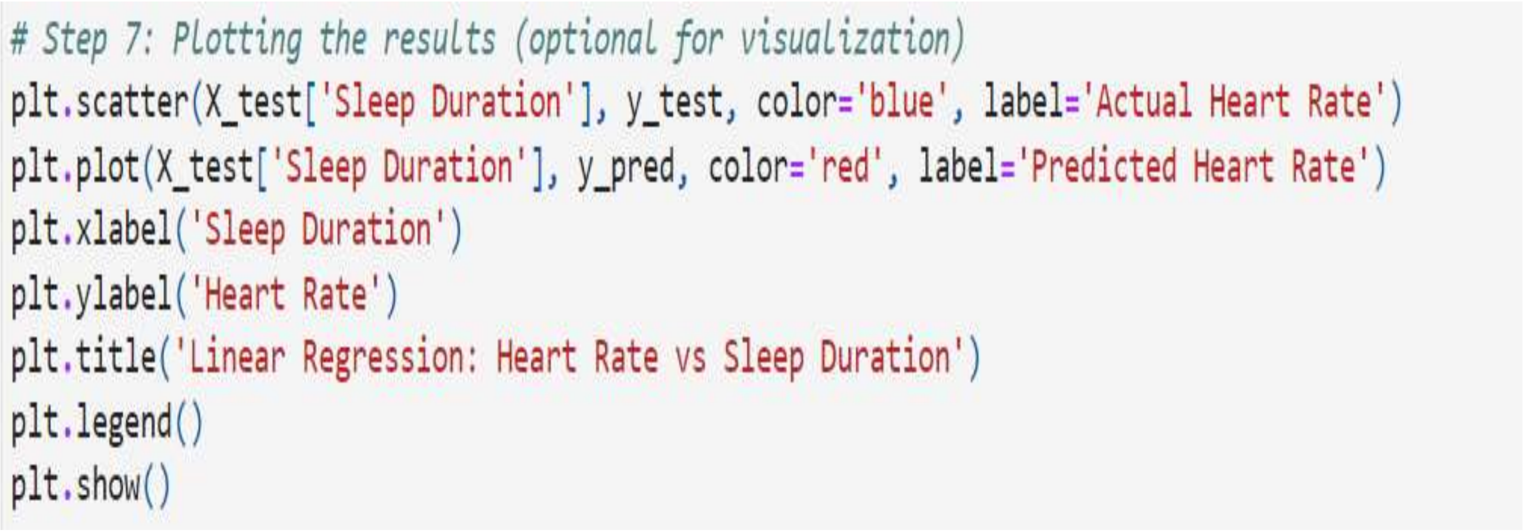
```
# Step 6: Coefficients of the linear regression (y = aX + b)
coefficients = lin_reg.coef_
intercept = lin_reg.intercept_
print(f"Coefficients: {coefficients}")
print(f"Intercept: {intercept}")
```

```
Coefficients: [-2.61662626]
Intercept: 88.7819237512695
```

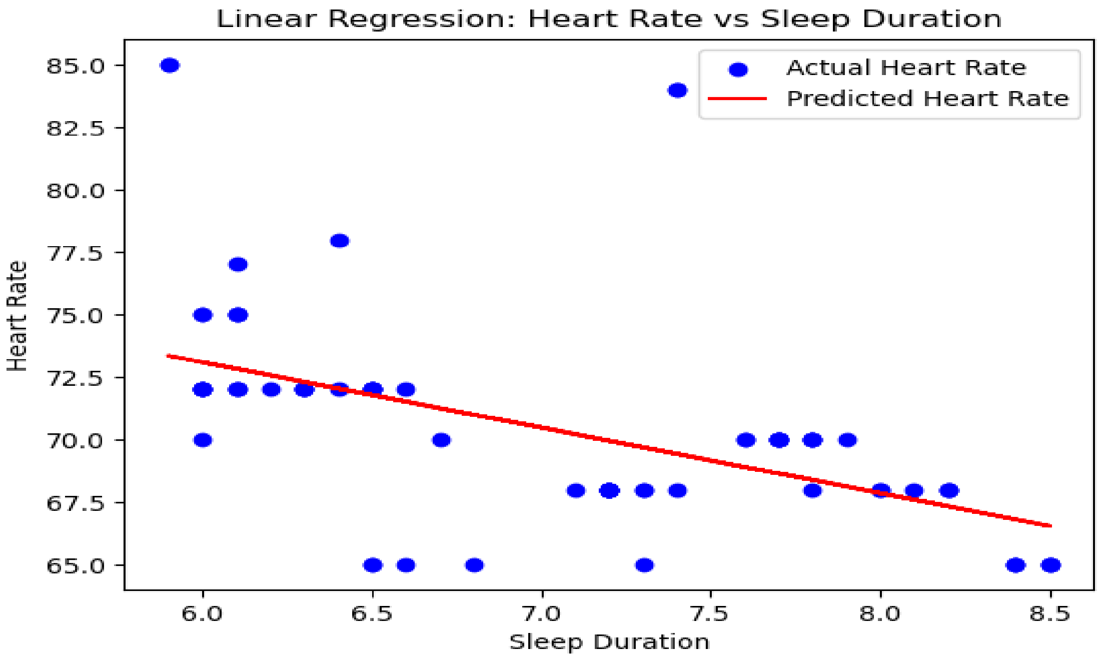
- The full regression equation based on your coefficients and intercept is:

$$\mathbf{y = -2.62X + 88.78}$$

Visualization



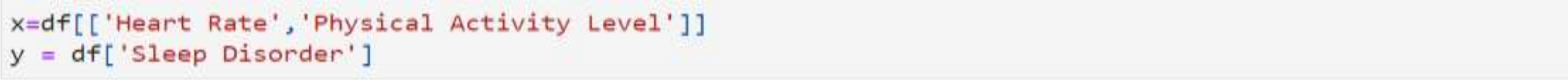
```
# Step 7: Plotting the results (optional for visualization)
plt.scatter(X_test['Sleep Duration'], y_test, color='blue', label='Actual Heart Rate')
plt.plot(X_test['Sleep Duration'], y_pred, color='red', label='Predicted Heart Rate')
plt.xlabel('Sleep Duration')
plt.ylabel('Heart Rate')
plt.title('Linear Regression: Heart Rate vs Sleep Duration')
plt.legend()
plt.show()
```


Linear Regression: Heart Rate vs Sleep Duration
● Actual Heart Rate
Predicted Heart Rate
Heart Rate
85.0
82.5
80.0
77.5
75.0
72.5
70.0
67.5
65.0
6.0
6.5
7.0
7.5
8.0
8.5
Sleep Duration

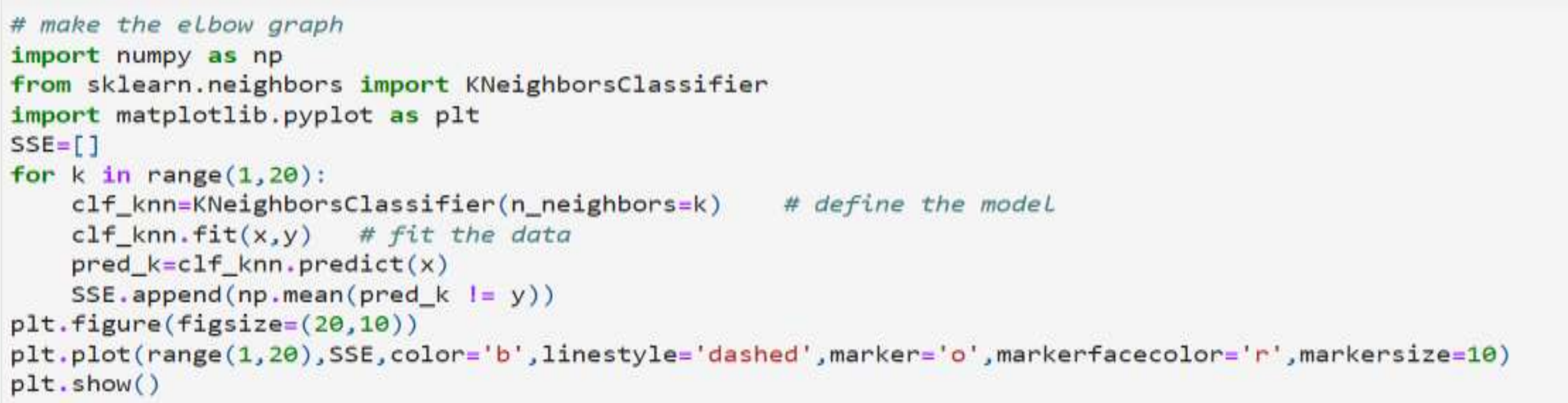
Interpretation

- The red line slopes downward, indicating a negative relationship between sleep duration and heart rate. This suggests that as sleep duration increases, the predicted heart rate decreases slightly.
- The model captures the general negative trend between sleep duration and heart rate.
- But the scatter of data points suggests that the linear model is not a perfect fit for predicting heart rate based on sleep duration alone.
- The linear regression suggests that longer sleep is associated with lower heart rate, but the scatter in actual values suggests that other factors contribute to heart rate variability, making sleep duration just one part of the equation.

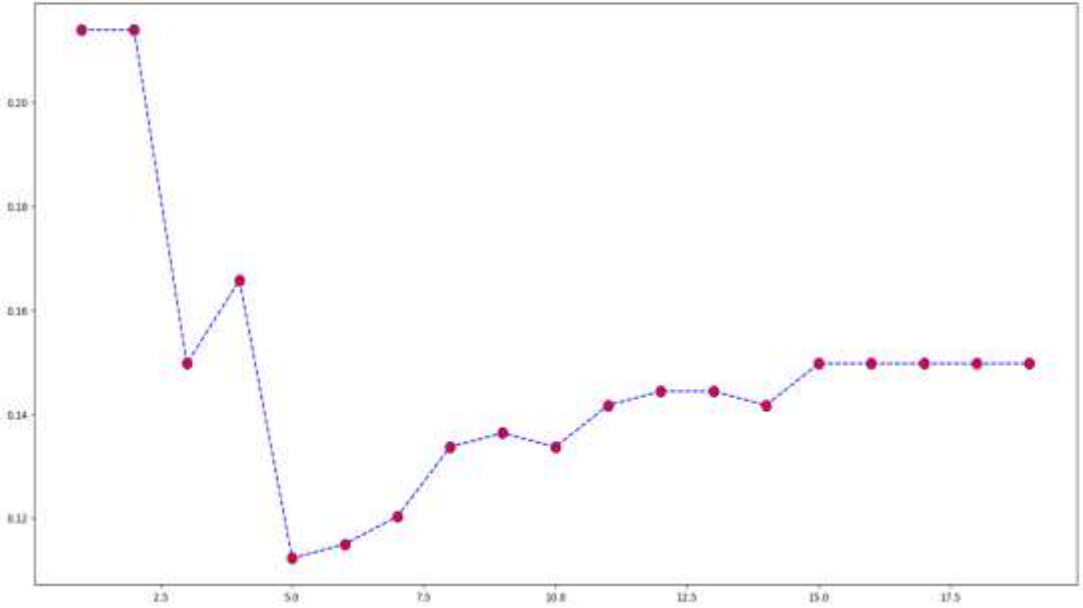
KNN (K-Nearest Neighbour)



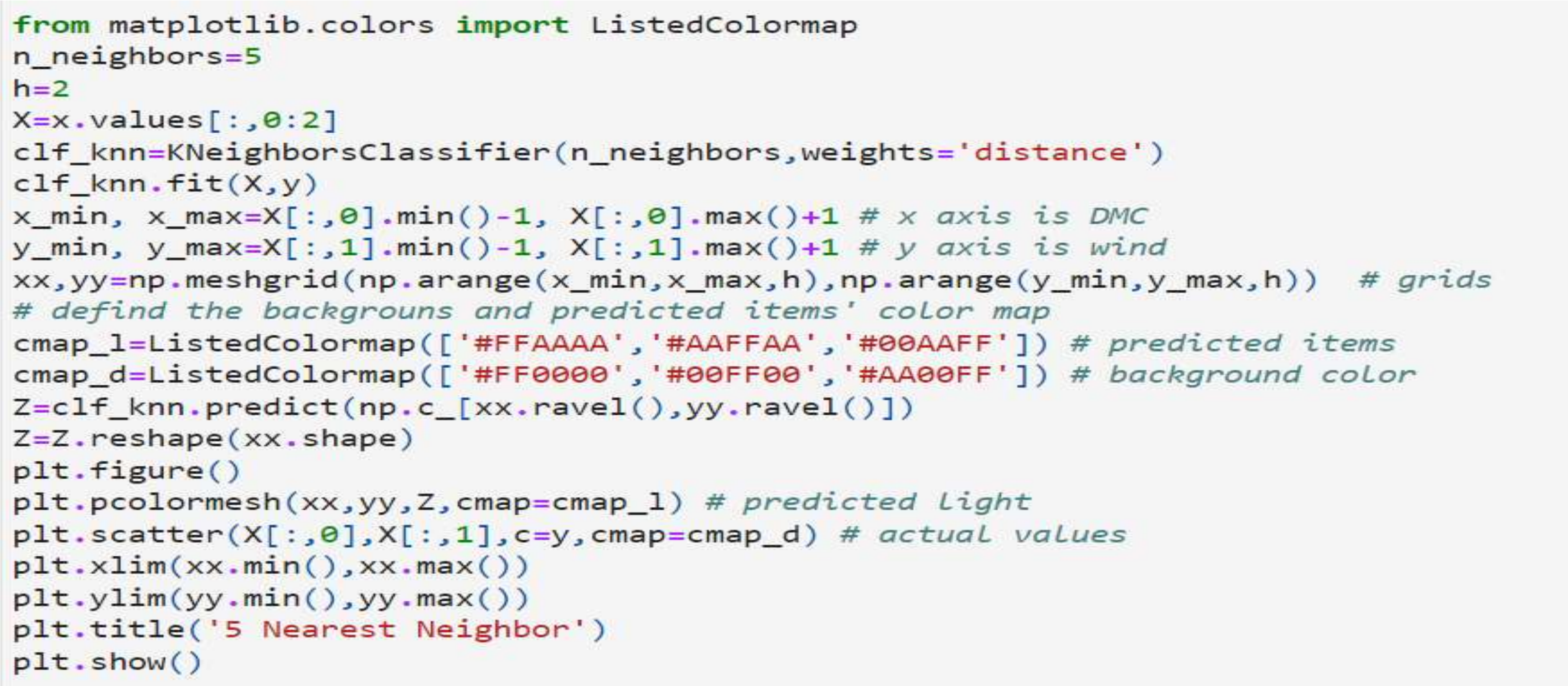
```
x=df[['Heart Rate','Physical Activity Level']]
y = df['Sleep Disorder']
```



```
# make the elbow graph
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
SSE=[]
for k in range(1,20):
    clf_knn=KNeighborsClassifier(n_neighbors=k)    # define the model
    clf_knn.fit(x,y)   # fit the data
    pred_k=clf_knn.predict(x)
    SSE.append(np.mean(pred_k != y))
plt.figure(figsize=(20,10))
plt.plot(range(1,20),SSE,color='b',linestyle='dashed',marker='o',markerfacecolor='r',markersize=10)
plt.show()
```

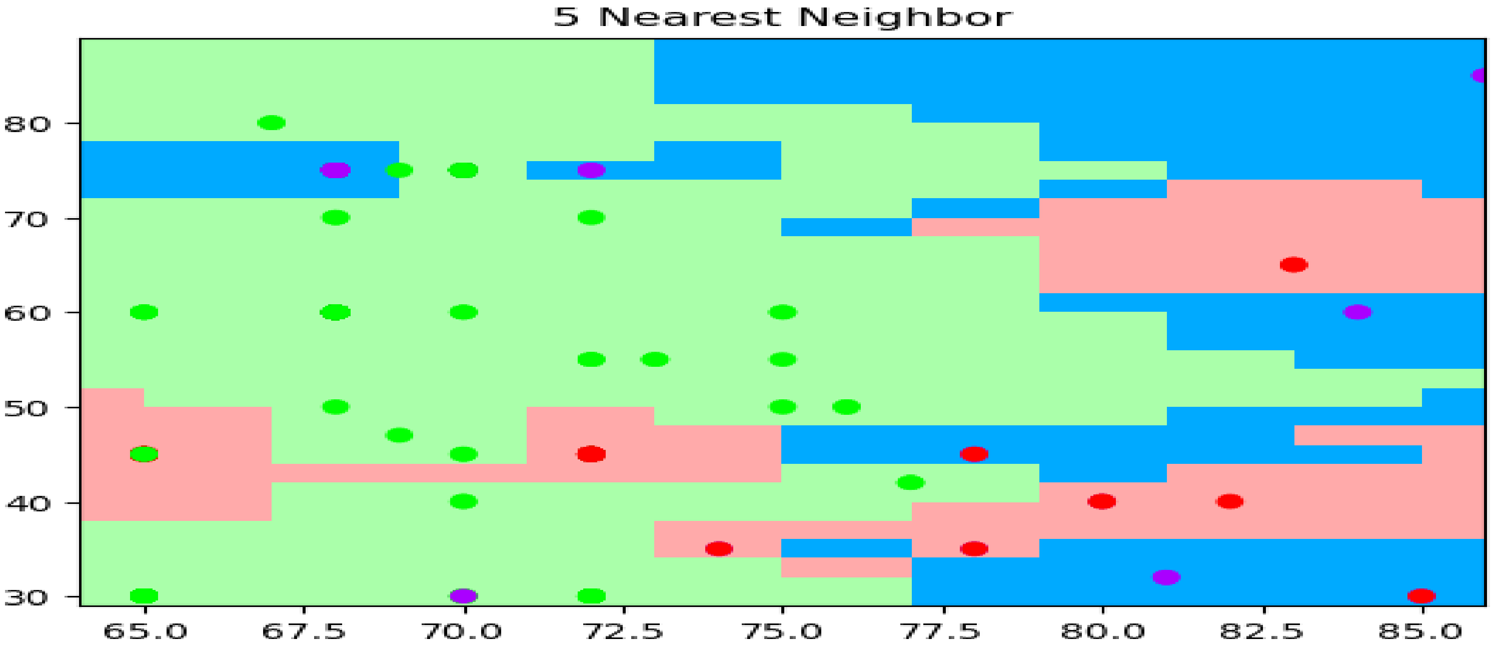


- From the plot , K=5.It shows a sharp drop from k = 1 to k = 5, and after k = 5, the slope starts to flatten out.
- This is called the Elbow point.
- based on the shape of this curve, k = 5 seems to be the optimal value.

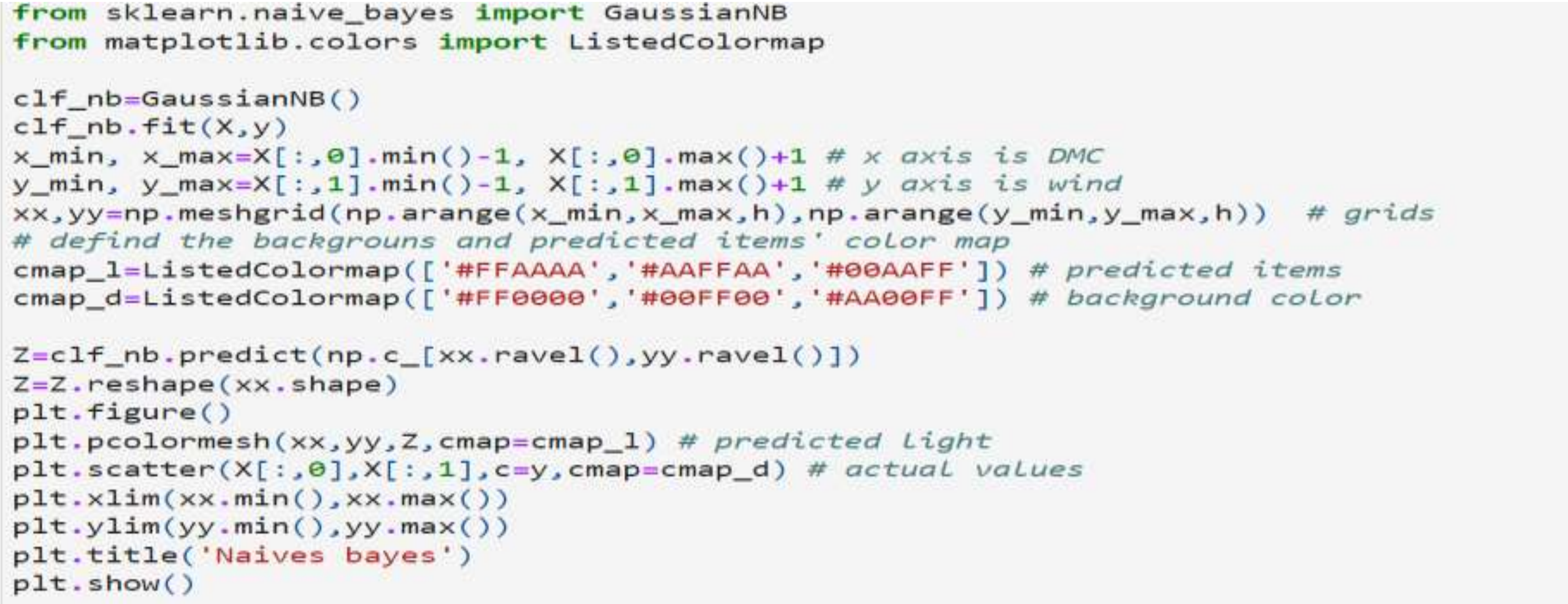


```
from matplotlib.colors import ListedColormap
n_neighbors=5
h=2
X=x.values[:,0:2]
clf_knn=KNeighborsClassifier(n_neighbors,weights='distance')
clf_knn.fit(X,y)
x_min, x_max=X[:,0].min()-1, X[:,0].max()+1 # x axis is DMC
y_min, y_max=X[:,1].min()-1, X[:,1].max()+1 # y axis is wind
xx,yy=np.meshgrid(np.arange(x_min,x_max,h),np.arange(y_min,y_max,h))  # grids
# defind the backgrouns and predicted items' color map
cmap_l=ListedColormap(['#FFAAAA','#AAFFAA','#00AAFF']) # predicted items
cmap_d=ListedColormap(['#FF0000','#00FF00','#AA00FF']) # background color
Z=clf_knn.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx,yy,Z,cmap=cmap_l) # predicted light
plt.scatter(X[:,0],X[:,1],c=y,cmap=cmap_d) # actual values
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title('5 Nearest Neighbor')
plt.show()
```

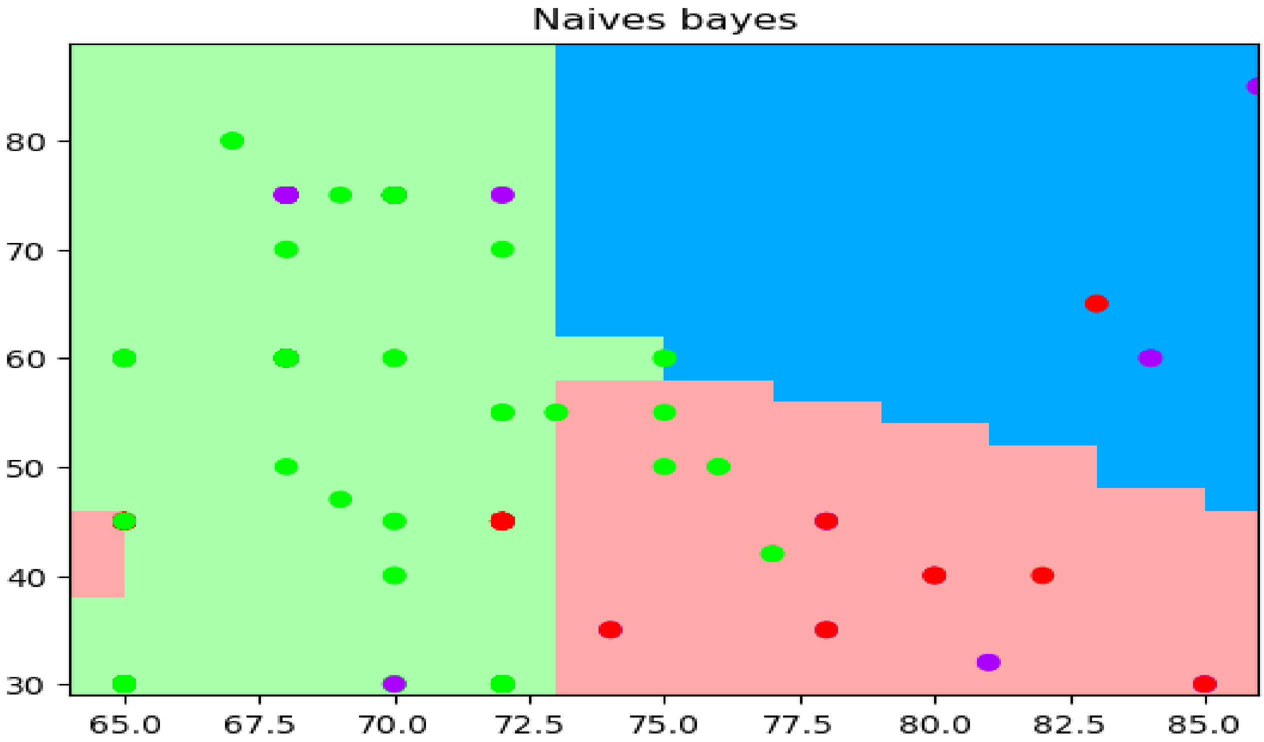
KNN plot


5 Nearest Neighbor
80
70
60
50
40
30
65.0
67.5
70.0
72.5
75.0
77.5
80.0
82.5
85.0

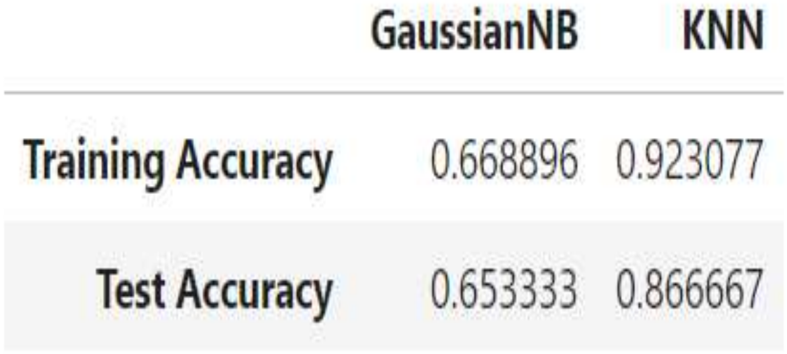
NB (Naive Bayes)



```
from sklearn.naive_bayes import GaussianNB
from matplotlib.colors import ListedColormap

clf_nb=GaussianNB()
clf_nb.fit(X,y)
x_min, x_max=X[:,0].min()-1, X[:,0].max()+1 # x axis is DMC
y_min, y_max=X[:,1].min()-1, X[:,1].max()+1 # y axis is wind
xx,yy=np.meshgrid(np.arange(x_min,x_max,h),np.arange(y_min,y_max,h))  # grids
# defind the backgrouns and predicted items' color map
cmap_l=ListedColormap(['#FFAAAA','#AAFFAA','#00AAFF']) # predicted items
cmap_d=ListedColormap(['#FF0000','#00FF00','#AA00FF']) # background color

Z=clf_nb.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx,yy,Z,cmap=cmap_l) # predicted light
plt.scatter(X[:,0],X[:,1],c=y,cmap=cmap_d) # actual values
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title('Naives bayes')
plt.show()
```


Naives bayes
80
70
60
50
40
30
65.0
67.5
70.0
72.5
75.0
77.5
80.0
82.5
85.0

Accuracies



	GaussianNB	KNN
Training Accuracy	0.668896	0.923077
Test Accuracy	0.653333	0.866667

Gaussian Naive Bayes:

Moderate accuracy: Naive Bayes has testing and training accuracies that fall between 60 and 70 percent. The model does rather well on both the training and test data, as evidenced by this modest degree of accuracy. The Naive Bayes model appears to generalize reasonably well, meaning it does not overfit to the training data, as indicated by the narrow difference between the training accuracy (66.89%) and testing accuracy (65.33%). It works similarly with test data that isn't visible.

K-Nearest Neighbors (KNN):

High training accuracy: On the training set, the KNN model achieves a very high accuracy of 92.31%. This implies that a large portion of the training data's complexity and patterns are being captured by KNN. Test accuracy slightly declines: KNN's test accuracy of 86.67% is still high but lower than the training accuracy. The model may be learning some patterns in the training data that don't transfer as well to the test set, which could be the cause of the KNN's test accuracy being marginally lower than its training accuracy. However, this overfitting isn't severe because the test accuracy is still fairly high.

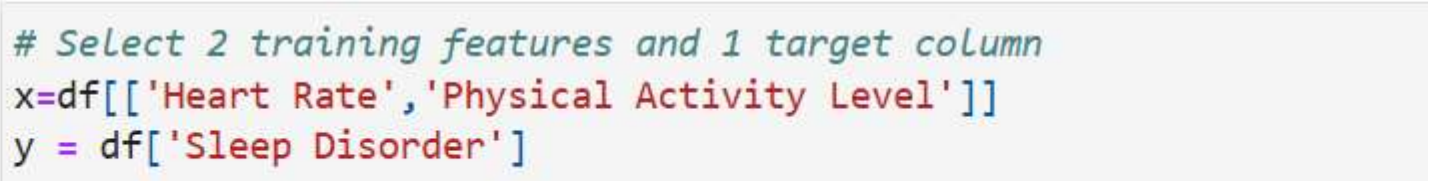
What model works better?

- Gaussian NB:This model has moderate accuracy and appears to generalize fairly well.
- KNN has higher accuracy but shows signs of potential overfitting.
- In general, KNN is performing better in terms of accuracy compared to Naive Bayes, but it may come at the cost of some overfitting, while Naive Bayes provides a more balanced, though less accurate, model.Based on the accuracies KNN is the best model.

SVM & KERNELS

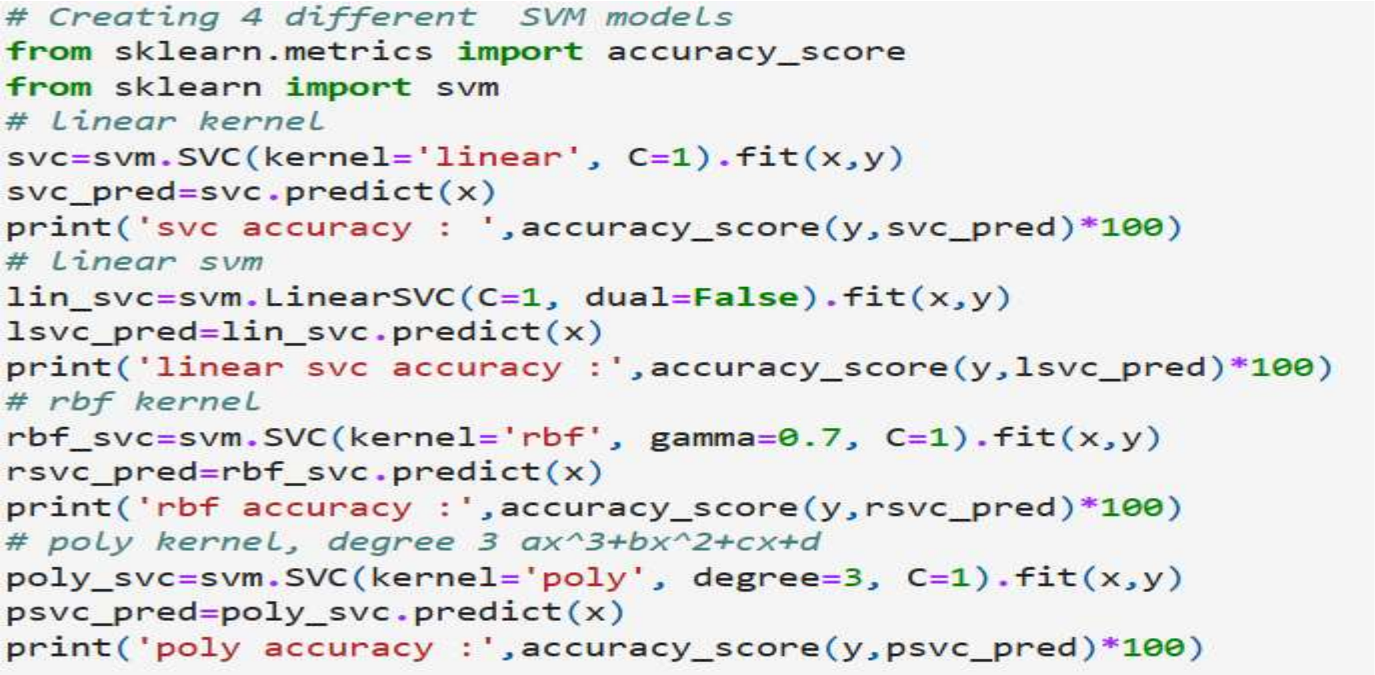
Selecting 2 training and 1 target variables:

Heart Rate and Physical Activity Level as training features and Sleep Disorder as target column



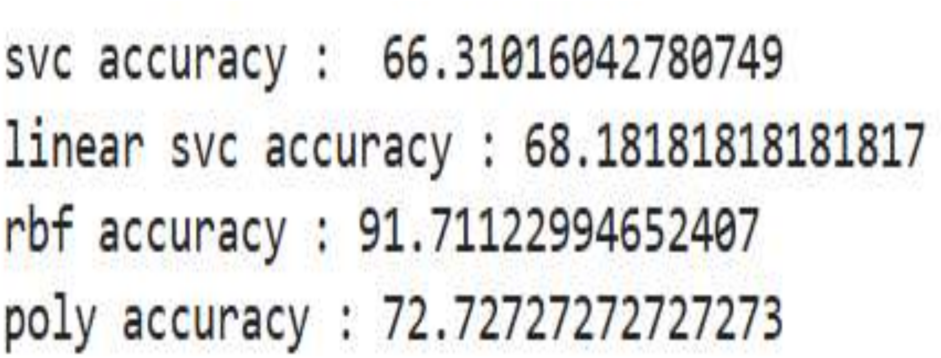
```
# Select 2 training features and 1 target column
x=df[['Heart Rate','Physical Activity Level']]
y = df['Sleep Disorder']
```

SVM models



```
# Creating 4 different  SVM models
from sklearn.metrics import accuracy_score
from sklearn import svm
# linear kernel
svc=svm.SVC(kernel='linear', C=1).fit(x,y)
svc_pred=svc.predict(x)
print('svc accuracy : ',accuracy_score(y,svc_pred)*100)
# linear svm
lin_svc=svm.LinearSVC(C=1, dual=False).fit(x,y)
lsvc_pred=lin_svc.predict(x)
print('linear svc accuracy :',accuracy_score(y,lsvc_pred)*100)
# rbf kernel
rbf_svc=svm.SVC(kernel='rbf', gamma=0.7, C=1).fit(x,y)
rsvc_pred=rbf_svc.predict(x)
print('rbf accuracy :',accuracy_score(y,rsvc_pred)*100)
# poly kernel, degree 3 ax^3+bx^2+cx+d
poly_svc=svm.SVC(kernel='poly', degree=3, C=1).fit(x,y)
psvc_pred=poly_svc.predict(x)
print('poly accuracy :',accuracy_score(y,psvc_pred)*100)
```

Accuracies:

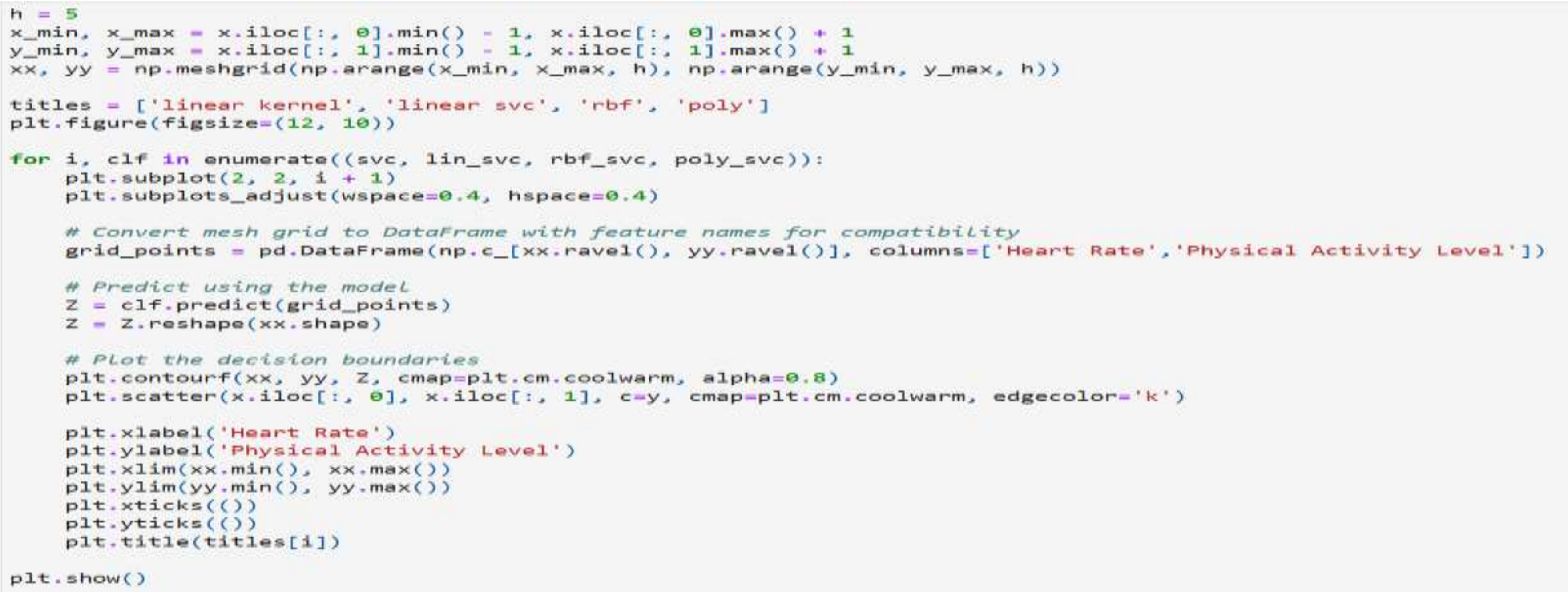


```
svc accuracy :  66.31016042780749
linear svc accuracy : 68.18181818181817
rbf accuracy : 91.71122994652407
poly accuracy : 72.72727272727273
```

Interpretation for accuracies

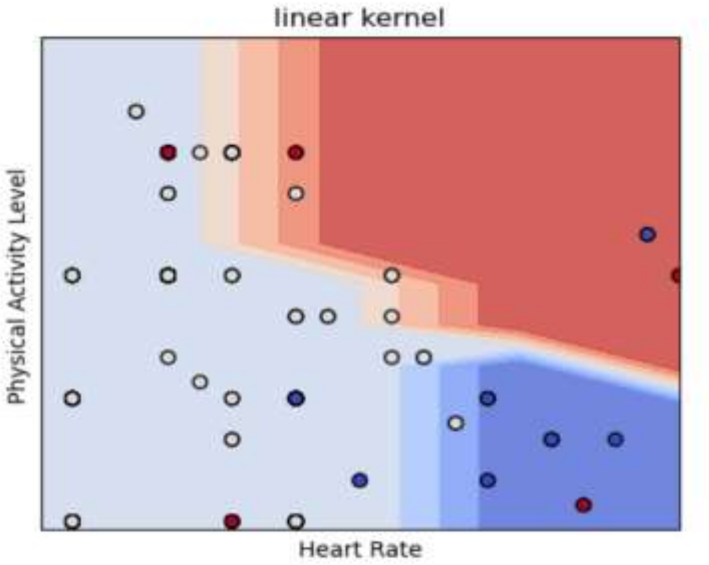
- **Linear SVM (66.31% Accuracy)**
  - Limited performance with a simple linear boundary.
  - Linear relationship does not sufficiently capture data patterns.
- **LinearSVC (68.18% Accuracy)**
  - Slightly better than Linear SVM but still limited.
  - Not suited for complex data patterns.
- **RBF SVM (91.71% Accuracy)**
  - Significantly higher accuracy with a non-linear boundary.
  - Captures complex relationships effectively.
  - **Best-performing model** for this dataset.
- **Polynomial SVM (72.73% Accuracy)**
  - Better than linear models but less effective than RBF.
  - Some non-linear relationships captured, but limited flexibility.

Kernels



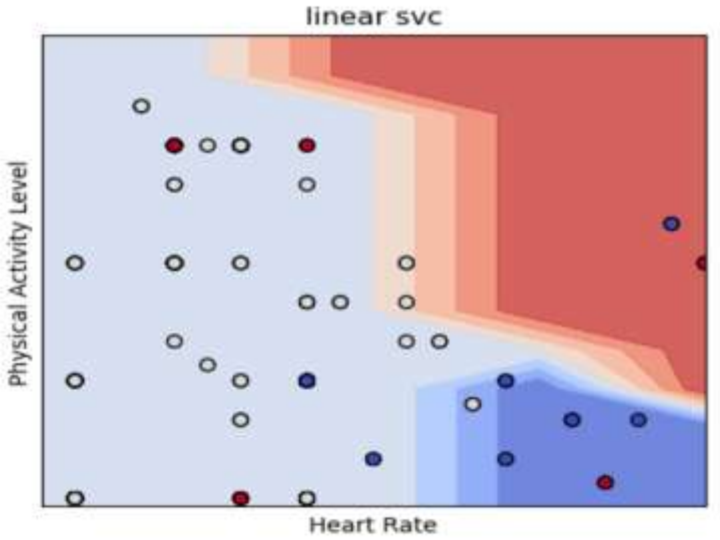
```
h = 5
x_min, x_max = x.iloc[:, 0].min() - 1, x.iloc[:, 0].max() + 1
y_min, y_max = x.iloc[:, 1].min() - 1, x.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

titles = ['linear kernel', 'linear svc', 'rbf', 'poly']
plt.figure(figsize=(12, 10))

for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    # Convert mesh grid to DataFrame with feature names for compatibility
    grid_points = pd.DataFrame(np.c_[xx.ravel(), yy.ravel()], columns=['Heart Rate','Physical Activity Level'])

    # Predict using the model
    Z = clf.predict(grid_points)
    Z = Z.reshape(xx.shape)

    # Plot the decision boundaries
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
    plt.scatter(x.iloc[:, 0], x.iloc[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolor='k')

    plt.xlabel('Heart Rate')
    plt.ylabel('Physical Activity Level')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```

Linear kernel



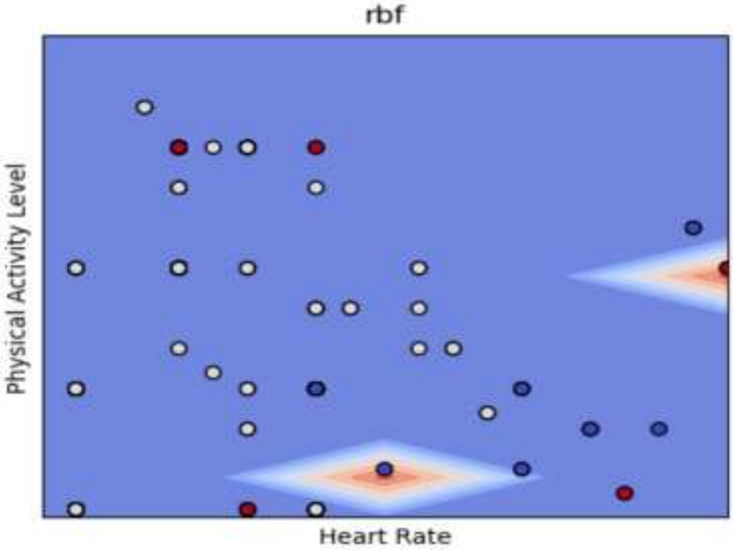
- ➢ A straight-line decision boundary shows limited flexibility, dividing the space into two primary regions (light blue and red).
- ➢ Misclassifications indicate it struggles to capture the actual data patterns, and its low accuracy highlights difficulty in handling non-linear relationships.

Linear SVC



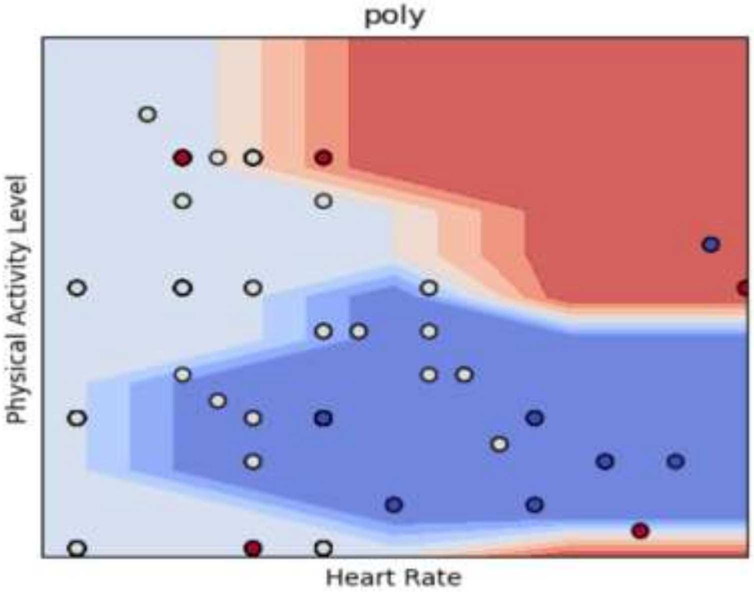
- ➢ A similar straight-line boundary with slight differences in positioning.
- ➢ Misclassifications occur due to limited ability to handle complex patterns. While accuracy is slightly better than the linear kernel, it remains constrained by its linear nature.

Rbf



- A complex, non-linear decision boundary with "islands" or "pockets" that effectively separates clusters and minimizes misclassification.
- The high accuracy reflects a strong capacity to capture intricate data patterns.

Poly

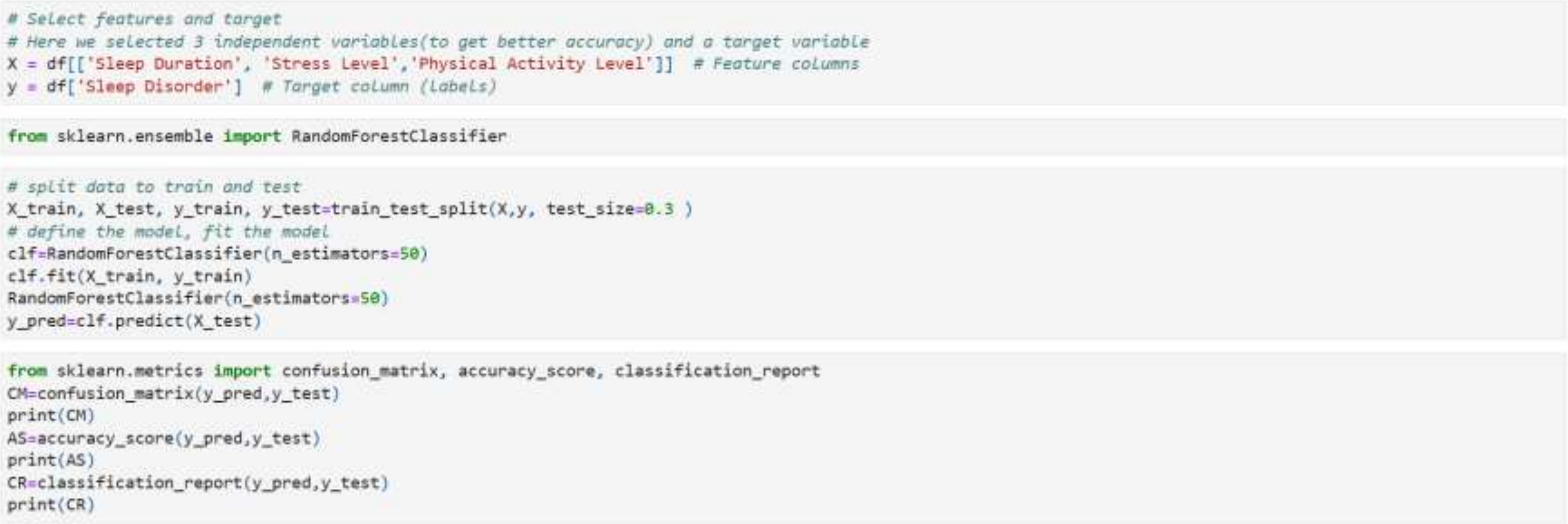


- ➢ A non-linear boundary with defined curvature, providing better separation than linear models but less adaptability than RBF.
- ➢ Shows moderate performance, capturing some non-linearity, though less effectively than RBF.

Best Performing Model:

RBF Kernel has the maximum accuracy because of its ability to capture complicated, non-linear interactions. Perfect for datasets with complex patterns that are too complex for linear models to handle. Limitations of Linear Models Low accuracy results from the straight-line boundaries produced by both linear kernel and linear SVC. Poor classification performance is the outcome of an inability to adjust to non-linear patterns. Perspectives on Polynomial Kernels offers some non-linear flexibility as a compromise. surpasses linear models in performance but falls short of the RBF kernel efficacy. Suggestion The RBF kernel is the suggested option for the best classification of data according to Age and Medication.

RANDOM FOREST



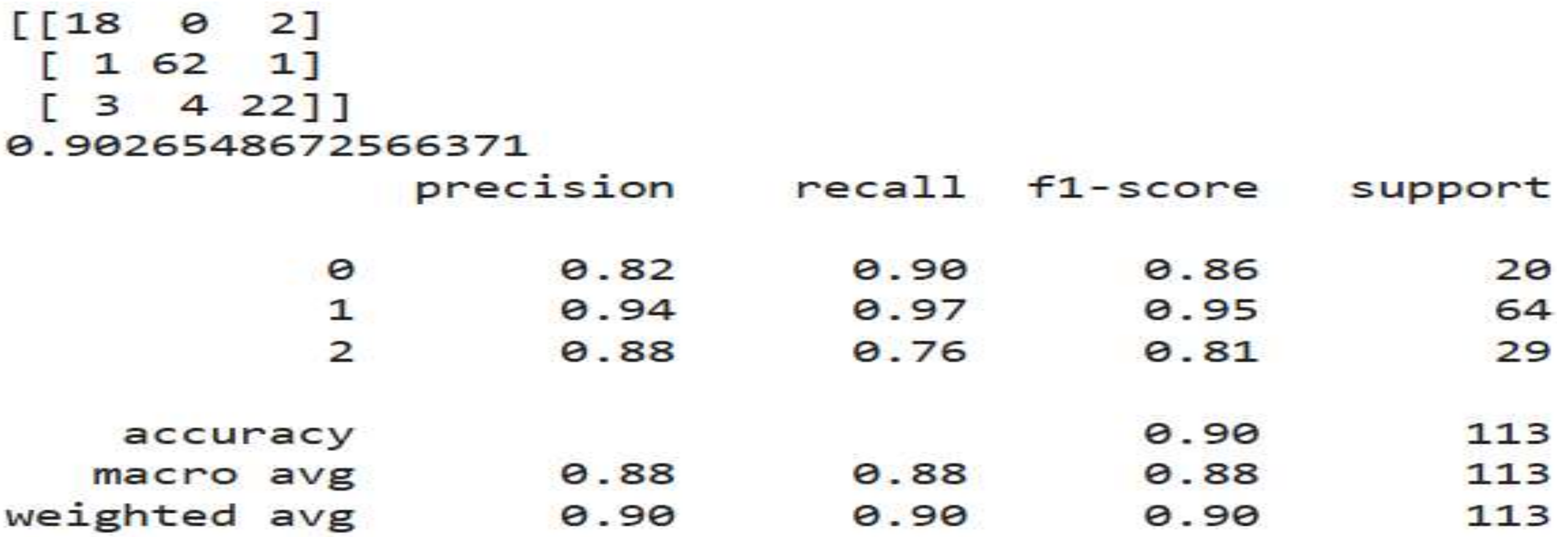
```
# Select features and target
# Here we selected 3 independent variables(to get better accuracy) and a target variable
X = df[['Sleep Duration', 'Stress Level','Physical Activity Level']]  # Feature columns
y = df['Sleep Disorder']  # Target column (labels)
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
# split data to train and test
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.3 )
# define the model, fit the model
clf=RandomForestClassifier(n_estimators=50)
clf.fit(X_train, y_train)
RandomForestClassifier(n_estimators=50)
y_pred=clf.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
CM=confusion_matrix(y_pred,y_test)
print(CM)
AS=accuracy_score(y_pred,y_test)
print(AS)
CR=classification_report(y_pred,y_test)
print(CR)
```

Output:

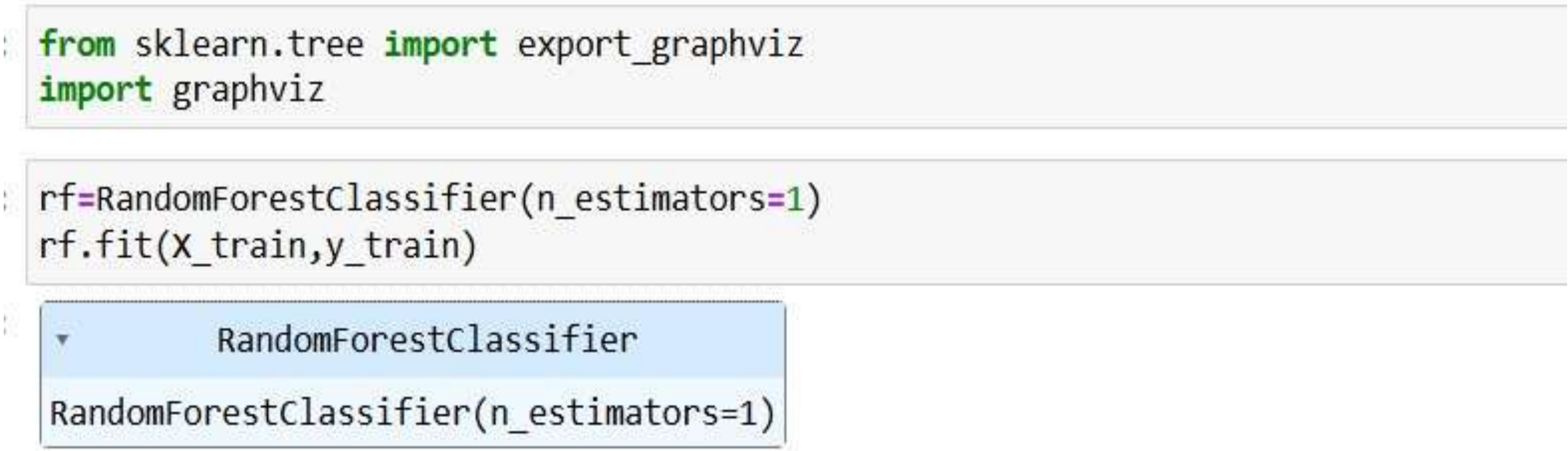


```
[[18  0  2]
 [ 1 62  1]
 [ 3  4 22]]
0.9026548672566371
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        20
           1       0.94      0.97      0.95        64
           2       0.88      0.76      0.81        29

    accuracy                           0.90       113
   macro avg       0.88      0.88      0.88       113
weighted avg       0.90      0.90      0.90       113
```

Interpretation:

The model scored an exceptional overall accuracy of 90.27%, with particularly strong performance in Class 1, which displayed both high precision (94%), and recall (97%). Class 0 likewise produced good results, with a precision of 82% and a recall of 90%. Class 2, while still good with 88% precision, trailed somewhat in recall at 76%, indicating room for growth. The model's effectiveness implies that, while it excels at detecting Class 1, improvements might be made to better distinguish Class 2, either by further tuning of model parameters or feature refinement to better discriminate between the classes.

Random forest



```
from sklearn.tree import export_graphviz
import graphviz
```

```
rf=RandomForestClassifier(n_estimators=1)
rf.fit(X_train,y_train)
```

```
▾          RandomForestClassifier

RandomForestClassifier(n_estimators=1)
```

Random forest tree: