Data_PreProcessing

October 11, 2024

0.0.1 Data PreProcessing

[1]:
```
#Import the necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder, OrdinalEncoder
from sklearn.compose import ColumnTransformer
```

[2]:
```
#Load the dataset
data = pd.read_csv('./data/Sleep_health_and_lifestyle_dataset.csv')
data.head()
```

[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	No
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	No
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	No
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea

[3]:
```
# Check for missing values
missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Person ID                  0
```

```
Gender                     0
Age                        0
Occupation                 0
Sleep Duration             0
Quality of Sleep           0
Physical Activity Level    0
Stress Level               0
BMI Category               0
Blood Pressure             0
Heart Rate                 0
Daily Steps                0
Sleep Disorder             0
dtype: int64
```

[4]:
```
#Encoding the data set
#Label encoding for Gender
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
```

[5]:
```
#One Hot Endcoding for Occupation
occupation_encoder = pd.get_dummies(data['Occupation'], prefix='Occupation')
data = pd.concat([data, occupation_encoder], axis=1)
data.drop('Occupation', axis=1, inplace=True)
```

[6]:
```
#Bmi Category
data['BMI Category'].unique()
```

[6]:
```
array(['Overweight', 'Normal', 'Obese', 'Normal Weight'], dtype=object)
```

[7]:
```
# We will replace Normal with Normal Weight
data['BMI Category']=data['BMI Category'].replace({'Normal':'Normal Weight'})
data['BMI Category'].unique()
```

[7]:
```
array(['Overweight', 'Normal Weight', 'Obese'], dtype=object)
```

[8]:
```
#Ordinal Encoding for BMI Category
bmi_category = ['Normal Weight', 'Overweight', 'Obese']
oe_bmi = OrdinalEncoder(categories=[bmi_category])
data['BMI Category'] = oe_bmi.fit_transform(data[['BMI Category']])
```

[9]:
```
#Spliting Blod Pressure into Systolic and Diastolic
data[['Systolic_BP', 'Diastolic_BP']] = data['Blood Pressure'].str.split('/',␣
 ↪expand=True)
data['Systolic_BP'] = data['Systolic_BP'].astype(int)
data['Diastolic_BP'] = data['Diastolic_BP'].astype(int)
data.drop('Blood Pressure', axis=1, inplace=True)
```

[10]:
```
#Label Encoding for Sleep Disorder
data['Sleep Disorder'] = label_encoder.fit_transform(data['Sleep Disorder'])
```

[11]:
```
# Standardize numerical features
numerical_features = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical␣
 ↪Activity Level',
                      'Stress Level', 'Systolic_BP', 'Diastolic_BP', 'Heart␣
 ↪Rate', 'Daily Steps']

ct = ColumnTransformer([
    ('scaler', StandardScaler(), numerical_features)
])

# Fit and transform the data
data[numerical_features] = ct.fit_transform(data[numerical_features])
```

[12]:
```
#print the head of the data
data.head()
```

[12]:

	Person ID	Gender	Age	Sleep Duration	Quality of Sleep	\
0	1	1	-1.753096	-1.298887	-1.098280	
1	2	1	-1.637643	-1.173036	-1.098280	
2	3	1	-1.637643	-1.173036	-1.098280	
3	4	1	-1.637643	-1.550588	-2.771424	
4	5	1	-1.637643	-1.550588	-2.771424	

	Physical Activity Level	Stress Level	BMI Category	Heart Rate	\
0	-0.825418	0.347021	1.0	1.654719	
1	0.039844	1.475592	0.0	1.170474	
2	0.039844	1.475592	0.0	1.170474	
3	-1.402260	1.475592	2.0	3.591698	
4	-1.402260	1.475592	2.0	3.591698	

	Daily Steps	…	Occupation_Lawyer	Occupation_Manager	Occupation_Nurse	\
0	-1.619584	…	False	False	False	
1	1.970077	…	False	False	False	
2	1.970077	…	False	False	False	
3	-2.362273	…	False	False	False	
4	-2.362273	…	False	False	False	

	Occupation_Sales Representative	Occupation_Salesperson	\
0	False	False	
1	False	False	
2	False	False	
3	True	False	
4	True	False	

```
   Occupation_Scientist  Occupation_Software Engineer  Occupation_Teacher  \
0                 False                          True               False
1                 False                         False               False
2                 False                         False               False
3                 False                         False               False
4                 False                         False               False

   Systolic_BP  Diastolic_BP
0    -0.330002     -0.268102
1    -0.459239     -0.755640
2    -0.459239     -0.755640
3     1.479309      0.869486
4     1.479309      0.869486

[5 rows x 24 columns]
```

[13]:
```
#Priint the columns of the data
data.columns
```

[13]:
```
Index(['Person ID', 'Gender', 'Age', 'Sleep Duration', 'Quality of Sleep',
       'Physical Activity Level', 'Stress Level', 'BMI Category', 'Heart Rate',
       'Daily Steps', 'Sleep Disorder', 'Occupation_Accountant',
       'Occupation_Doctor', 'Occupation_Engineer', 'Occupation_Lawyer',
       'Occupation_Manager', 'Occupation_Nurse',
       'Occupation_Sales Representative', 'Occupation_Salesperson',
       'Occupation_Scientist', 'Occupation_Software Engineer',
       'Occupation_Teacher', 'Systolic_BP', 'Diastolic_BP'],
      dtype='object')
```

[14]:
```
#Drop the columns that are not needed
data.drop(['Person ID'], axis=1, inplace=True)

#Cleaned data
data.to_csv('./data/cleaned_data.csv', index=False)
```

[15]:
```
#Split the data into features and target
X = data.drop('Sleep Disorder', axis=1)
y = data['Sleep Disorder']
```

[16]:
```
#Split the data into training and testing sets and Save the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
#Save the data
X_train.to_csv('./data/X_train.csv', index=False)
X_test.to_csv('./data/X_test.csv', index=False)
y_train.to_csv('./data/y_train.csv', index=False)
y_test.to_csv('./data/y_test.csv', index=False)
```