BMI_KNN_MuhammadMuzaki_21538141024

November 1, 2023

[187]:
```
# Muhammad Muzaki
# 21538141024

# NB : index
# 0: Extremly Weak
# 1:Weak
# 2 :Normal
# 3:Overweight
# 4:Obesity
# 5:Extremly Obesity
```

[189]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[190]:
```
df_train = pd.read_csv("bmi_train.csv")
df_validate = pd.read_csv("bmi_validation.csv")
```

[192]:
```
df_train.shape, df_validate.shape
```

[192]:
```
((400, 4), (100, 3))
```

[194]:
```
df_train
```

[194]:

	Gender	Height	Weight	Index
0	Male	161	89	4
1	Male	179	127	4
2	Male	172	139	5
3	Male	153	104	5
4	Male	165	68	2
..	…	…	…	…
395	Male	166	160	5
396	Male	145	130	5
397	Male	178	138	5
398	Male	168	158	5
399	Male	161	155	5

```
[400 rows x 4 columns]
```

[196]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  400 non-null    object
 1   Height  400 non-null    int64
 2   Weight  400 non-null    int64
 3   Index   400 non-null    int64
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

[197]:
```
df_train.isnull().sum()
```

[197]:
```
Gender    0
Height    0
Weight    0
Index     0
dtype: int64
```

[199]:
```
df_train.duplicated().sum()
```

[199]:
```
8
```

[200]:
```
df_train.drop_duplicates(keep='first', inplace=True)
```

[203]:
```
df_train.shape
```

[203]:
```
(392, 4)
```

[205]:
```
df_train['Gender'].unique()
```

[205]:
```
array(['Male', 'Female'], dtype=object)
```

[207]:
```
df_train['Gender'].value_counts()
```

[207]:
```
Gender
Male      197
Female    195
Name: count, dtype: int64
```

[209]:
```
df_train['Gender'] = df_train['Gender'].replace({'Male':0, 'Female':1})
```

[211]:
```
df_train.describe()
```

[211]:

	Gender	Height	Weight	Index
count	392.000000	392.000000	392.000000	392.000000
mean	0.497449	170.339286	106.224490	3.737245
std	0.500632	16.615701	32.510012	1.379366
min	0.000000	140.000000	50.000000	0.000000
25%	0.000000	156.000000	80.000000	3.000000
50%	0.000000	171.000000	107.000000	4.000000
75%	1.000000	184.000000	137.000000	5.000000
max	1.000000	199.000000	160.000000	5.000000

[215]:
```
df_train
```

[215]:

	Gender	Height	Weight	Index
0	0	161	89	4
1	0	179	127	4
2	0	172	139	5
3	0	153	104	5
4	0	165	68	2
..	…	…	…	…
395	0	166	160	5
396	0	145	130	5
397	0	178	138	5
398	0	168	158	5
399	0	161	155	5

[392 rows x 4 columns]

[217]:
```
X = df_train.drop('Index', axis=1)
Y = df_train['Index']
```

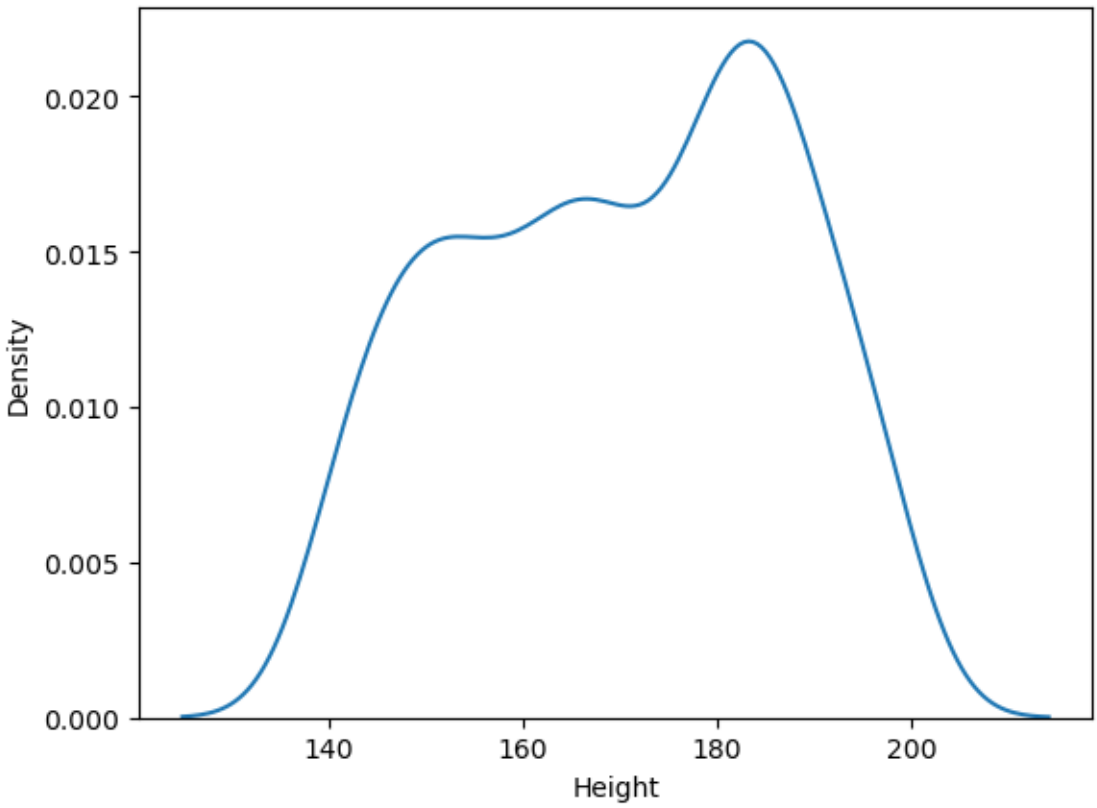
[218]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X, Y, test_size=0.20,␣
 ↪random_state=42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[218]:
```
((313, 3), (79, 3), (313,), (79,))
```

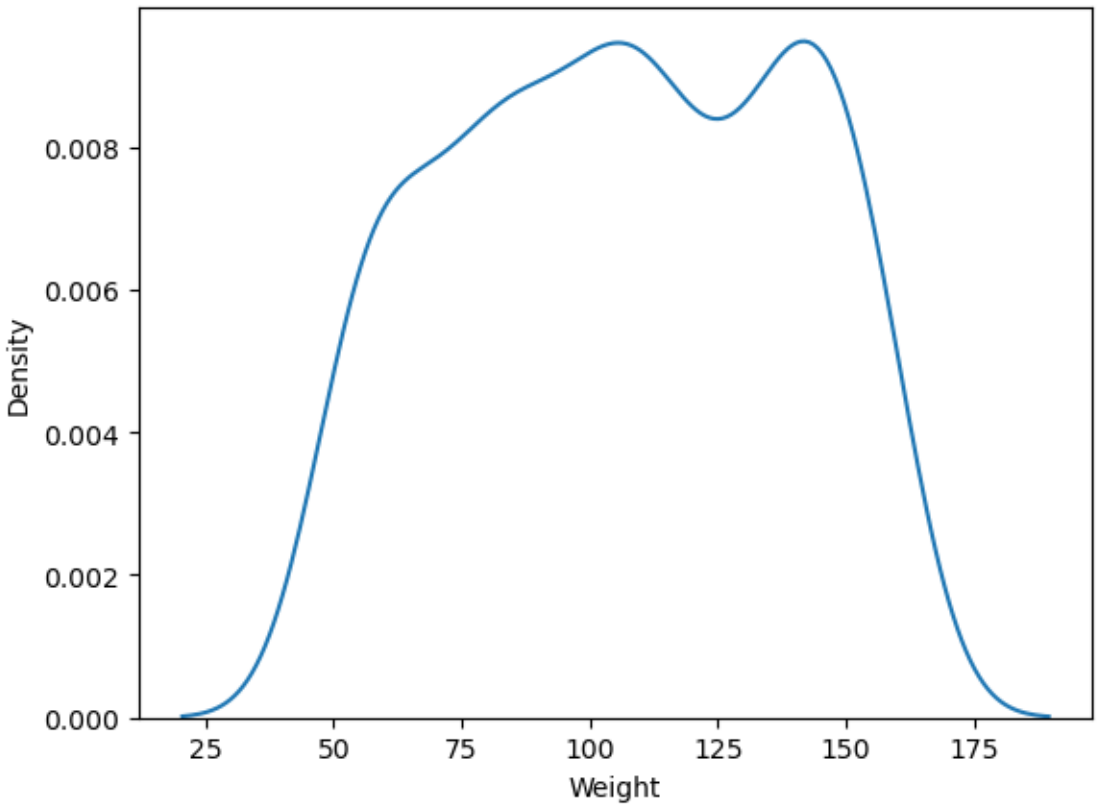
[221]:
```
sns.kdeplot(x=df_train['Height'])
```

[221]:
```
<Axes: xlabel='Height', ylabel='Density'>
```

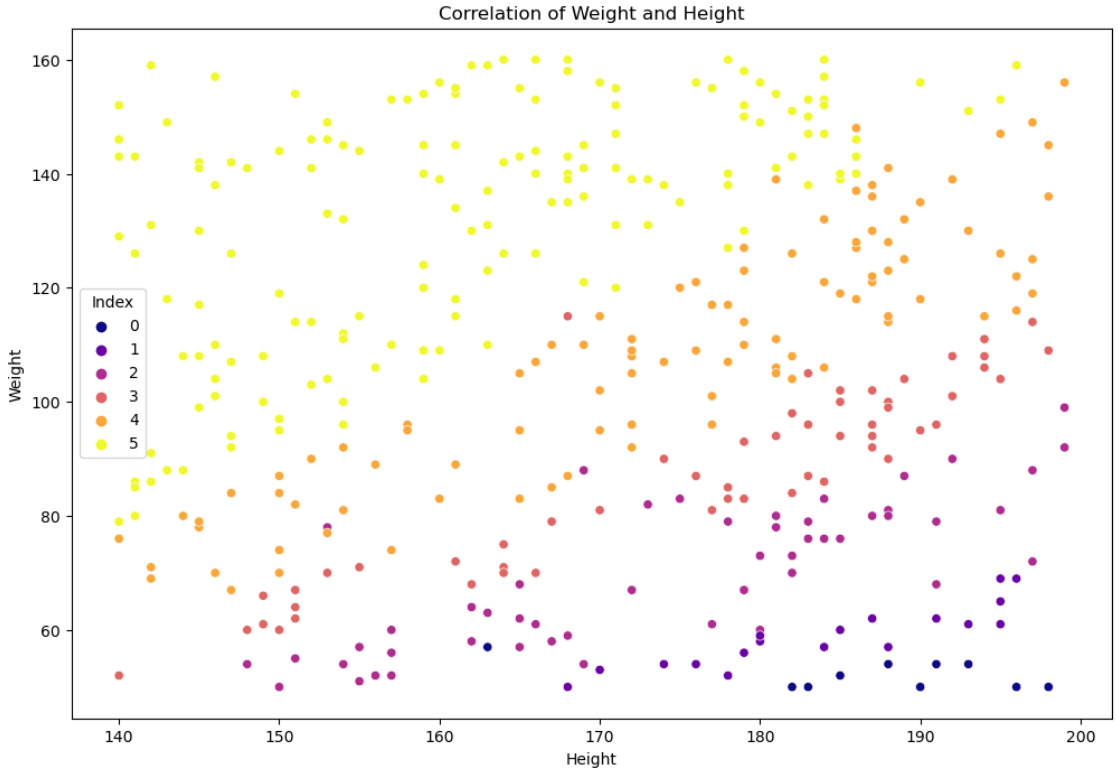


[222]:
```
sns.kdeplot(x=df_train['Weight'])
```

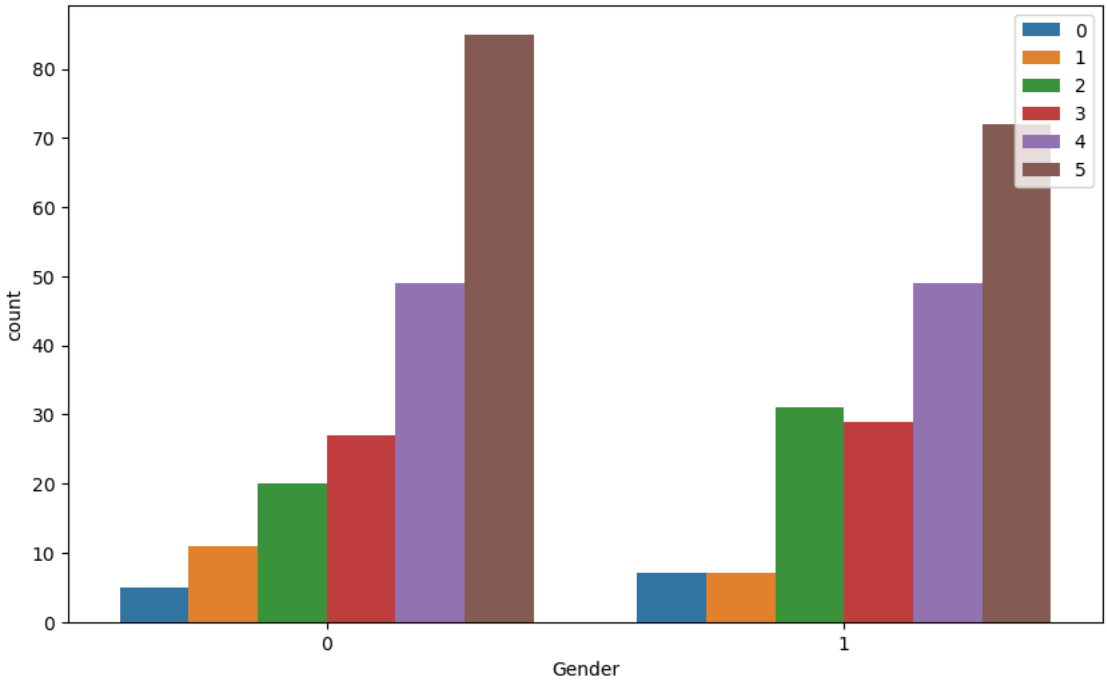
[222]:
```
<Axes: xlabel='Weight', ylabel='Density'>
```



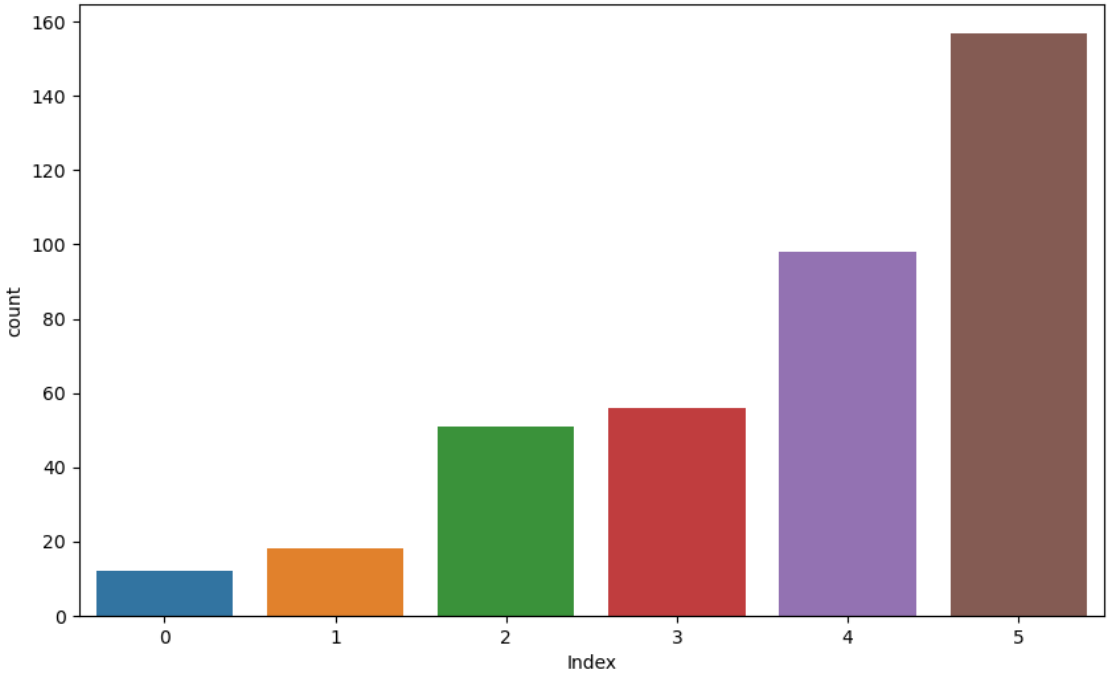
[225]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(data=df_train, x='Height', y='Weight', hue='Index',␣
↪palette='plasma')
plt.xlabel('Height')
plt.ylabel('Weight')
plt.title('Correlation of Weight and Height')
plt.show()
```



[226]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df_train, x = "Gender", hue='Index')
plt.legend(fontsize="small", ncol=3)
plt.legend(loc='upper right')
plt.show()
```



[229]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df_train, x = "Index")
plt.show()
```



[231]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[233]:
```
knn_clf = KNeighborsClassifier(n_neighbors=5, metric='euclidean', p=2)
knn_clf.fit(x_train, y_train)
```

[233]:
```
KNeighborsClassifier(metric='euclidean')
```

[235]:
```
y_pred = knn_clf.predict(x_test)
```

[237]:
```
pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
```

[237]:

	Actual	Predicted
78	5	4
278	0	0
248	5	5
55	5	5
395	5	5
..	…	…
369	2	3
82	4	4
115	0	0
3	5	5
18	5	5

[79 rows x 2 columns]

[239]:
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[241]:
```
score = accuracy_score(y_test, y_pred)
score
```

[241]:
```
0.8734177215189873
```

[243]:
```
matrix = confusion_matrix(y_test, y_pred)
matrix
```

[243]:
```
array([[ 3,  0,  0,  0,  0,  0],
       [ 0,  5,  1,  0,  0,  0],
       [ 0,  0,  8,  4,  0,  0],
       [ 0,  0,  0,  7,  0,  0],
       [ 0,  0,  0,  1, 15,  3],
       [ 0,  0,  0,  0,  1, 31]])
```

[245]:
```
report = classification_report(y_test, y_pred)
print(report)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	1.00	0.83	0.91	6
2	0.89	0.67	0.76	12
3	0.58	1.00	0.74	7
4	0.94	0.79	0.86	19
5	0.91	0.97	0.94	32
accuracy			0.87	79
macro avg	0.89	0.88	0.87	79
weighted avg	0.90	0.87	0.87	79

[ ]: