Obesity Dataset with k-Nearest Neighbors

Context

Welcome to the **Scaler Healthcare** data analysis team! As part of our ongoing efforts to understand and combat obesity globally, we're leveraging data to gain insights into factors contributing to obesity. You've been tasked with analyzing the Obesity Dataset, applying the k-Nearest Neighbors (kNN) algorithm to predict obesity levels based on individuals' eating habits and physical conditions.

Dataset Description

The dataset you'll be working with contains attributes related to individuals' eating habits and physical conditions. Here's a breakdown of the features you'll encounter:

Eating Habits Attributes:

1. **FAVC (Frequent consumption of high caloric food):** Indicates if the individual frequently eats high caloric food.
2. **FCVC (Frequency of consumption of vegetables):** Reflects how often the individual consumes vegetables.
3. **NCP (Number of main meals):** Represents the number of main meals the individual has in a day.
4. **CAEC (Consumption of food between meals):** Shows how frequently the individual eats between meals.
5. **CH20 (Consumption of water daily):** Details the daily water consumption.
6. **CALC (Consumption of alcohol):** Provides information on the individual's alcohol consumption.

Physical Condition Attributes:

7. **SCC (Calories consumption monitoring):** Indicates if the individual monitors their calorie intake.
8. **FAF (Physical activity frequency):** Reflects the frequency of physical activity.
9. **TUE (Time using technology devices):** Denotes the time spent using technology devices.
10. **MTRANS (Transportation used):** Details the primary mode of transportation.

```
# Downloading the Dataset
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/070/840/original/ObesityDataSet.csv
```

```
--2025-08-03 07:40:48--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/070/840/original/ObesityDataSet.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 3.167.84.28, 3.167.84.148, 3.167.84.9, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|3.167.84.28|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 263646 (257K) [text/plain]
Saving to: 'ObesityDataSet.csv'

ObesityDataSet.csv  100%[===================>] 257.47K  --.-KB/s    in 0.01s

2025-08-03 07:40:49 (22.1 MB/s) - 'ObesityDataSet.csv' saved [263646/263646]
```

```
import pandas as pd

df = pd.read_csv("ObesityDataSet.csv")
df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_

Next steps: Generate code with df | View recommended plots | New interactive sheet

KNN Lecture-1 Assesments

Analyzing Skewness in the Obesity Dataset

Context

You are tasked with exploring the Obesity dataset to understand the distribution of various features, especially focusing on skewness. This analysis is crucial for applying the kNN algorithm effectively.

Dataset

The dataset includes attributes related to individuals' eating habits and physical conditions. You'll examine these features to identify any right-skewness and explore the relationship between the **mean, median, and mode** for those features.

Task

Your goal is to perform Exploratory Data Analysis (EDA) to identify which of the given features are right-skewed. For any right-skewed feature, you should determine how the mean, median, and mode relate to each other.

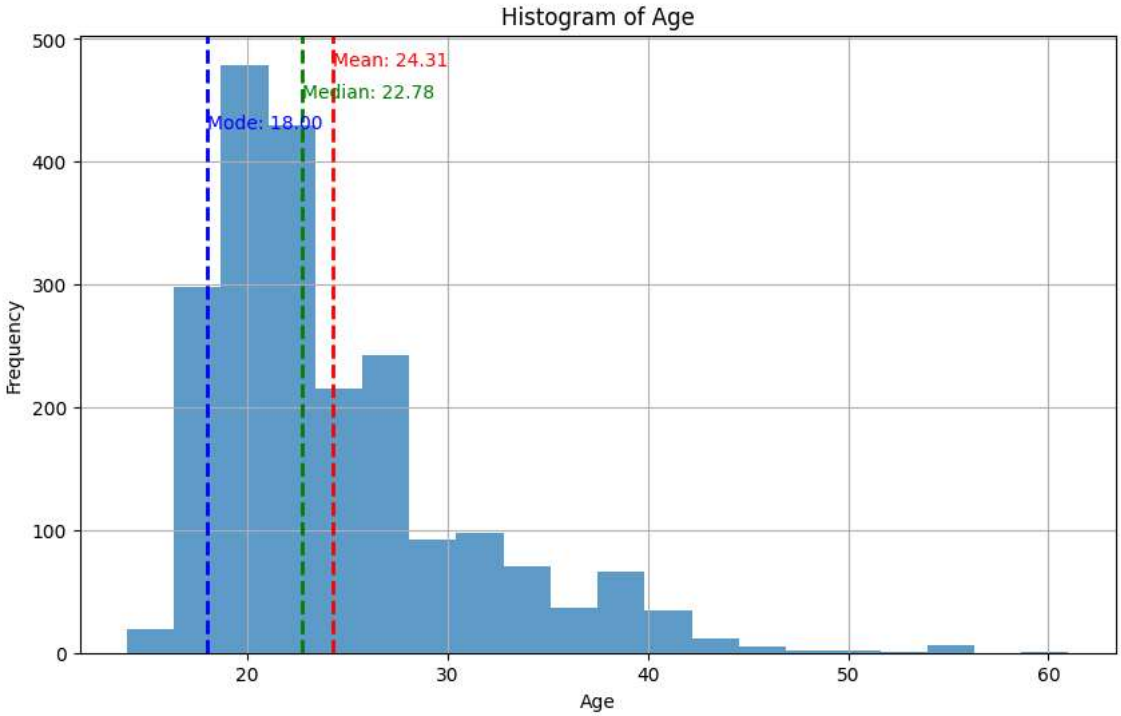
Instructions

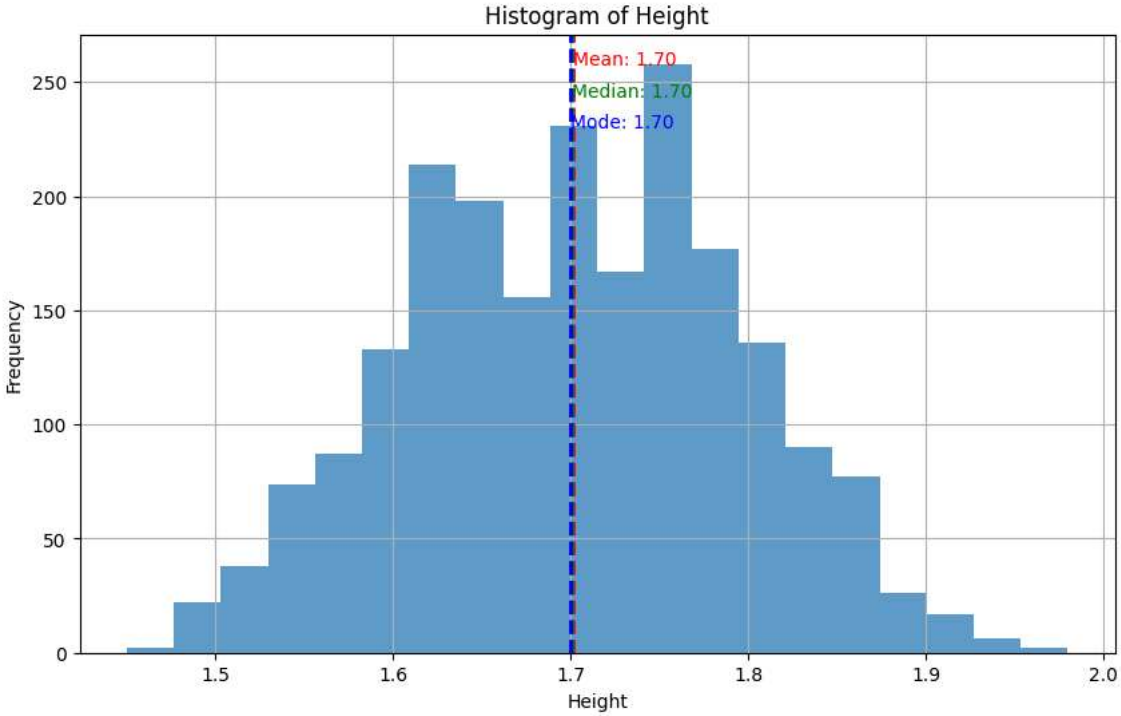
1. Load the dataset.
2. Conduct EDA focusing on the skewness of the attributes.
3. Identify any right-skewed features.
4. For each right-skewed feature, analyze the relationship between the mean, median, and mode.

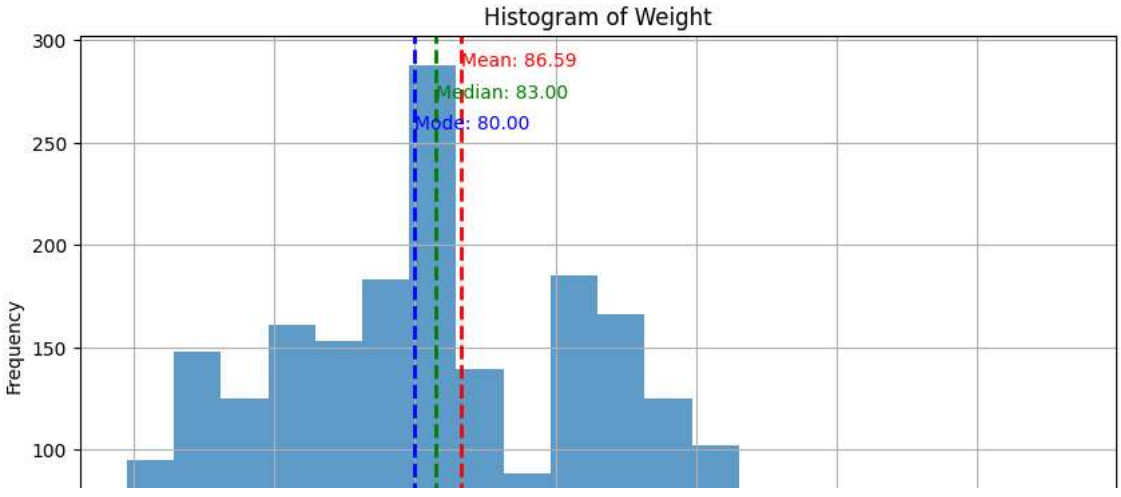
Complete the code template below to analyze the skewness of the features and understand the distribution of the dataset. Once you have identified the right-skewed features, calculate and print the mean, median, and mode for each to observe their relationship.

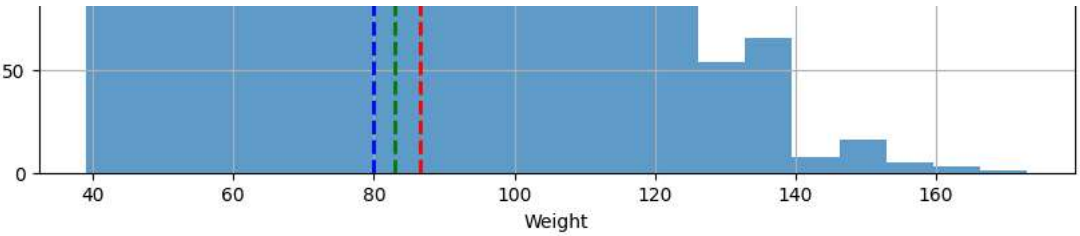
```
import numpy as np
import matplotlib.pyplot as plt

for feature in df.columns:
    # Skipping non-numeric features for histogram plotting
    if df[feature].dtype == "object":
        continue

    # Calculate mean, median, and mode
    mean = df[feature].mean()  # Fill in the blank to calculate mean
    median = df[feature].median()  # Fill in the blank to calculate median
    mode = df[feature].mode().get(0, np.nan)  # Fill in the blank to calculate mode

    # Create the histogram
    plt.figure(figsize=(10, 6))
    df[feature].hist(bins=20, alpha=0.7)

    # Add lines for mean, median, and mode
    plt.axvline(mean, color='red', linestyle='dashed', linewidth=2)  # Fill in the blank to plot mean
    plt.axvline(median, color='green', linestyle='dashed', linewidth=2)  # Fill in the blank to plot median
    plt.axvline(mode, color='blue', linestyle='dashed', linewidth=2)  # Fill in the blank to plot mode

    # Add annotations for mean, median, and mode
    plt.text(mean, plt.ylim()[1] * 0.95, f'Mean: {mean:.2f}', color = 'red')  # Fill in the blank to annotate mean
    plt.text(median, plt.ylim()[1] * 0.90, f'Median: {median:.2f}', color = 'green')  # Fill in the blank to annotate median
    plt.text(mode, plt.ylim()[1] * 0.85, f'Mode: {mode:.2f}', color = 'blue')  # Fill in the blank to annotate mode

    # Add title and labels
    plt.title(f'Histogram of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')

    # Show the plot
    plt.show()
```

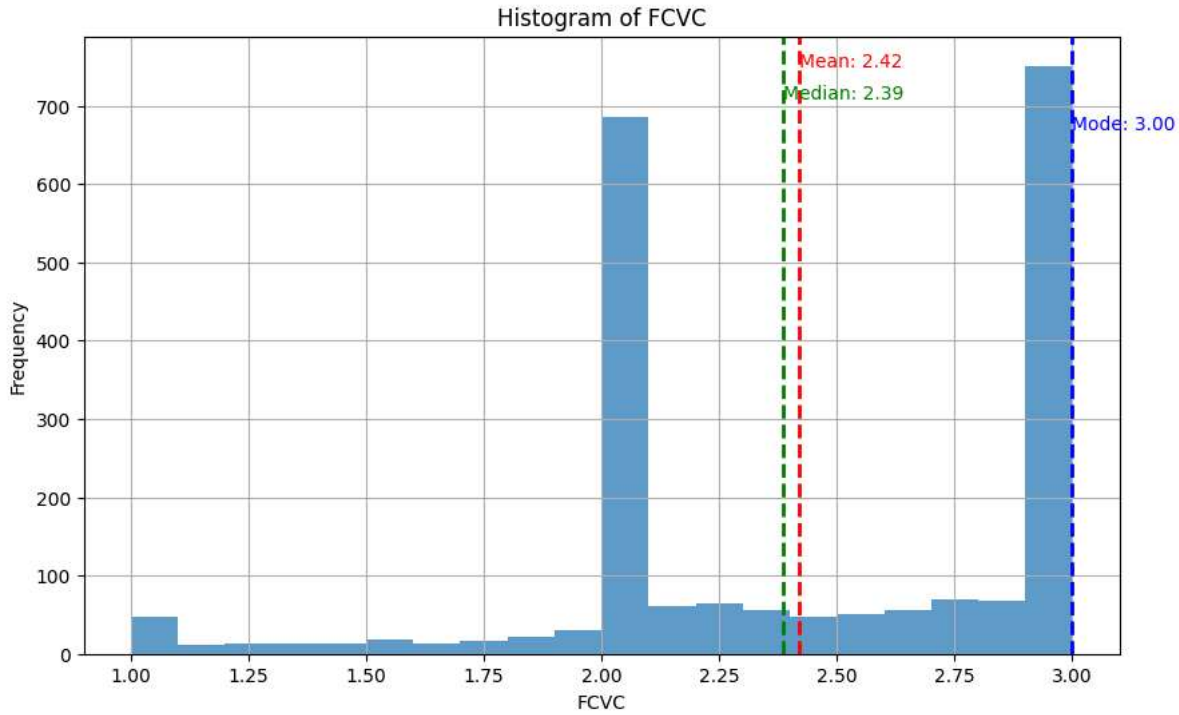
[4]

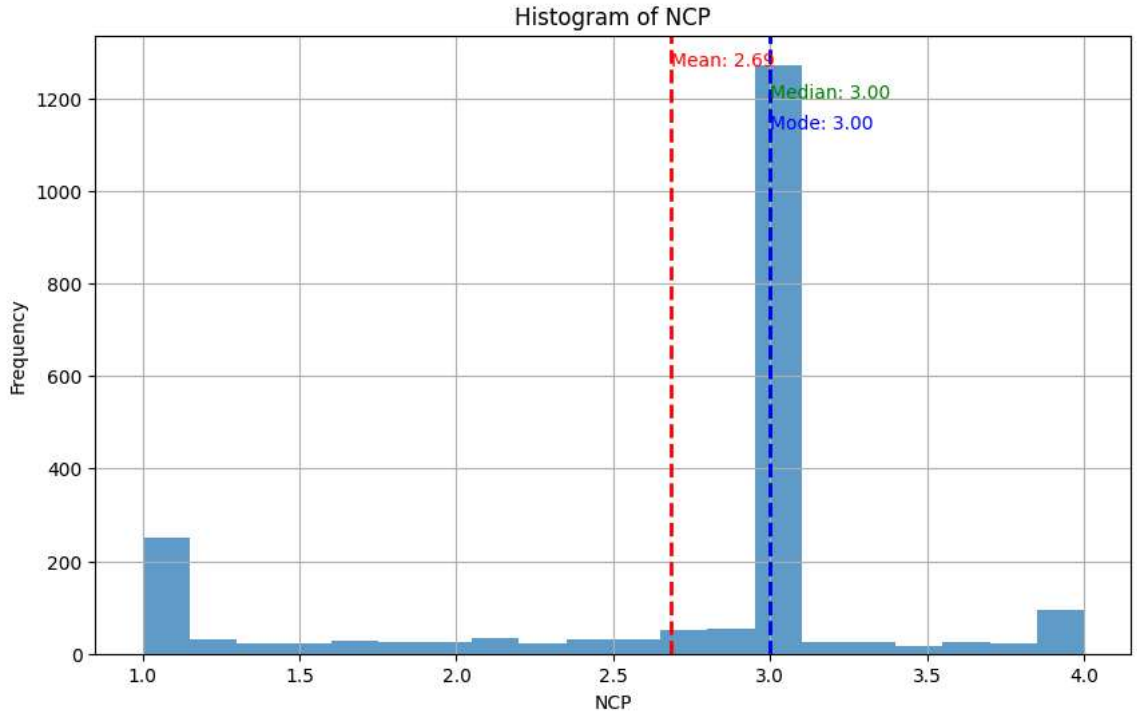

Histogram of Age
Mean: 24.31
Median: 22.78
Mode: 18.00
Frequency
Age
0
100
200
300
400
500
20
30
40
50
60

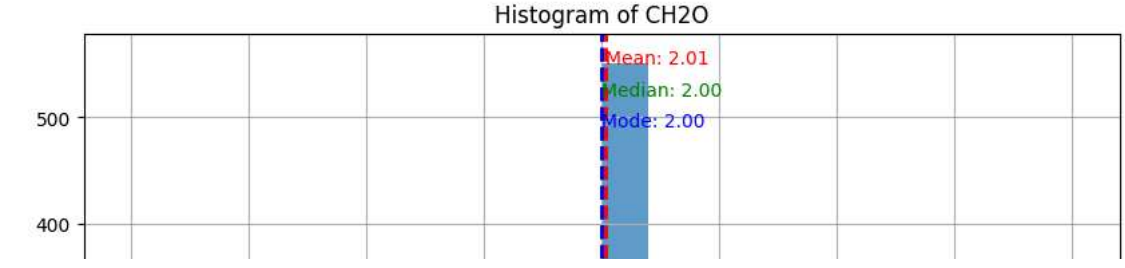

Histogram of Height
Mean: 1.70
Median: 1.70
Mode: 1.70
Frequency
Height
0
50
100
150
200
250
1.5
1.6
1.7
1.8
1.9
2.0

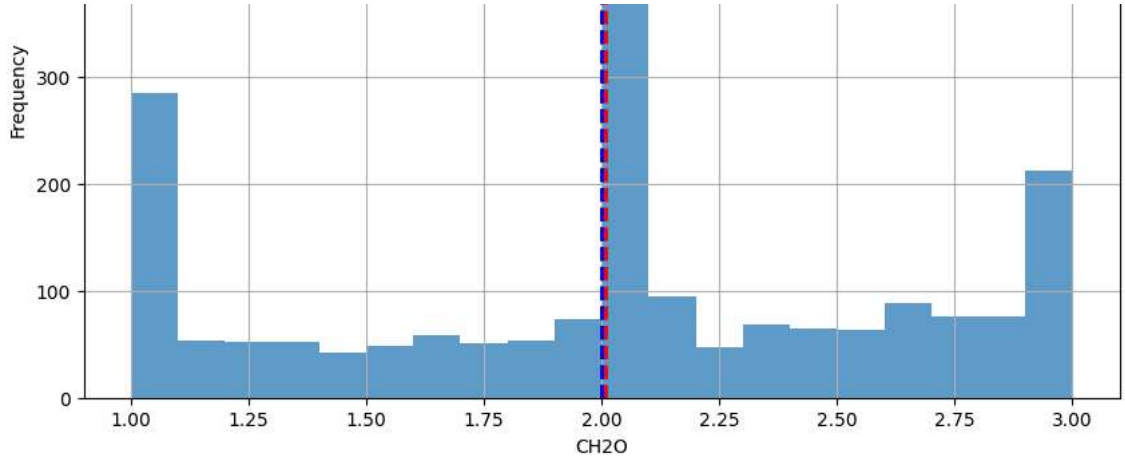

Histogram of Weight
Mean: 86.59
Median: 83.00
Mode: 80.00
Frequency
100
150
200
250
300

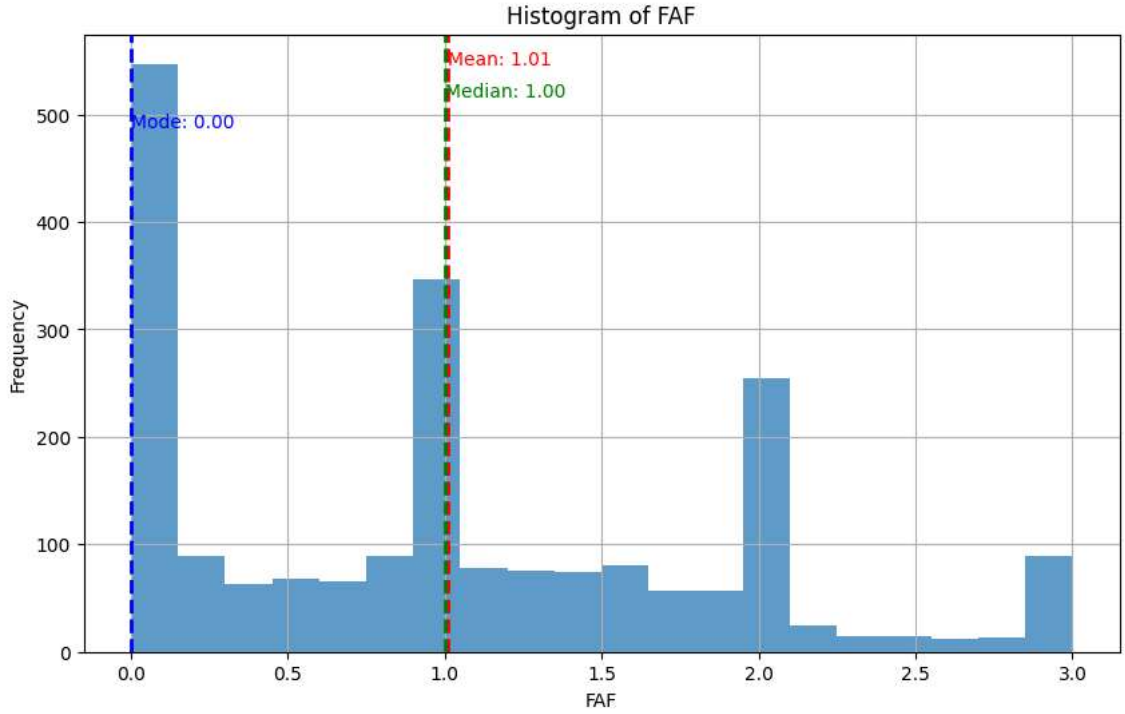

50
0
40
60
80
100
120
140
160
Weight

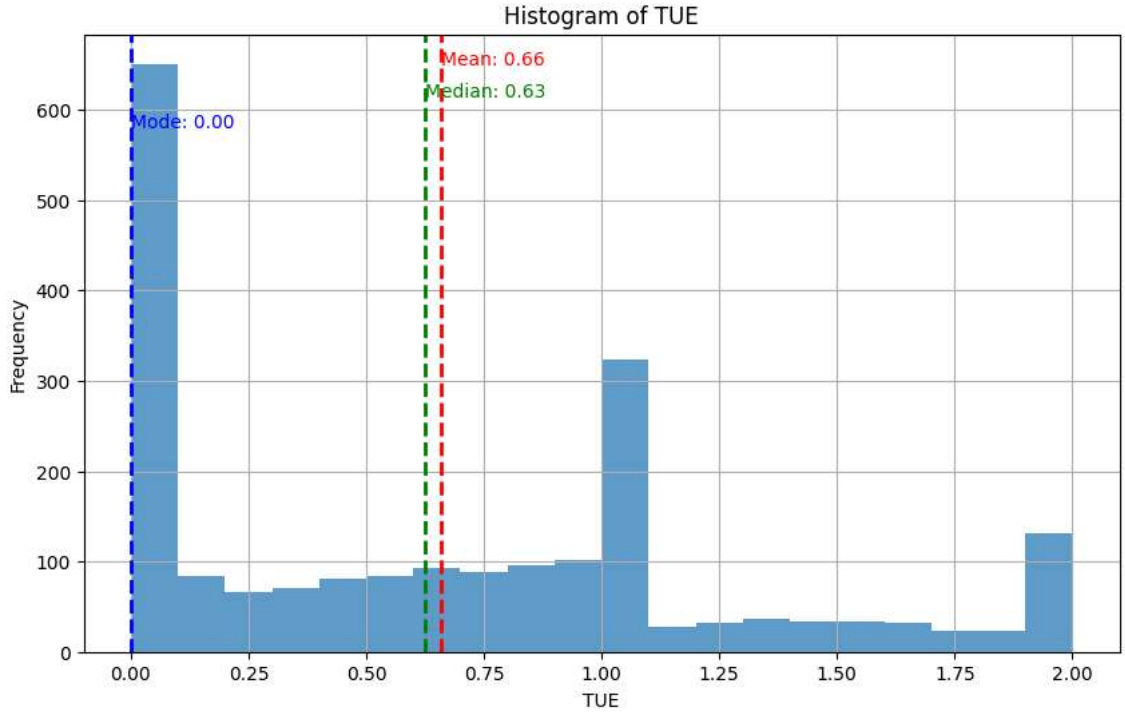

Histogram of FCVC
Mean: 2.42
Median: 2.39
Mode: 3.00
Frequency
700
600
500
400
300
200
100
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
FCVC


Histogram of NCP
Mean: 2.69
Median: 3.00
Mode: 3.00
Frequency
1200
1000
800
600
400
200
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
NCP


Histogram of CH2O
Mean: 2.01
Median: 2.00
Mode: 2.00
500
400


Frequency
300
200
100
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
CH2O


Histogram of FAF
Mean: 1.01
Median: 1.00
Mode: 0.00
Frequency
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
FAF


Histogram of TUE
Mean: 0.66
Median: 0.63
Mode: 0.00
Frequency
600
500
400
300
200
100
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
TUE

Analyzing Feature Interactions in the Obesity Dataset

Context:

After exploring individual feature distributions in the Obesity dataset, your next objective is to examine the interactions between these features. Understanding the relationships between different features can provide insights into the dataset's structure and inform the feature selection process, especially for machine learning algorithms that are sensitive to multicollinearity, such as kNN.

Task:

Your task is to create and analyze a correlation matrix for the numerical features in the Obesity dataset. This matrix will help you identify how different features are related and determine if there are any pairs of features that exhibit a high degree of correlation.

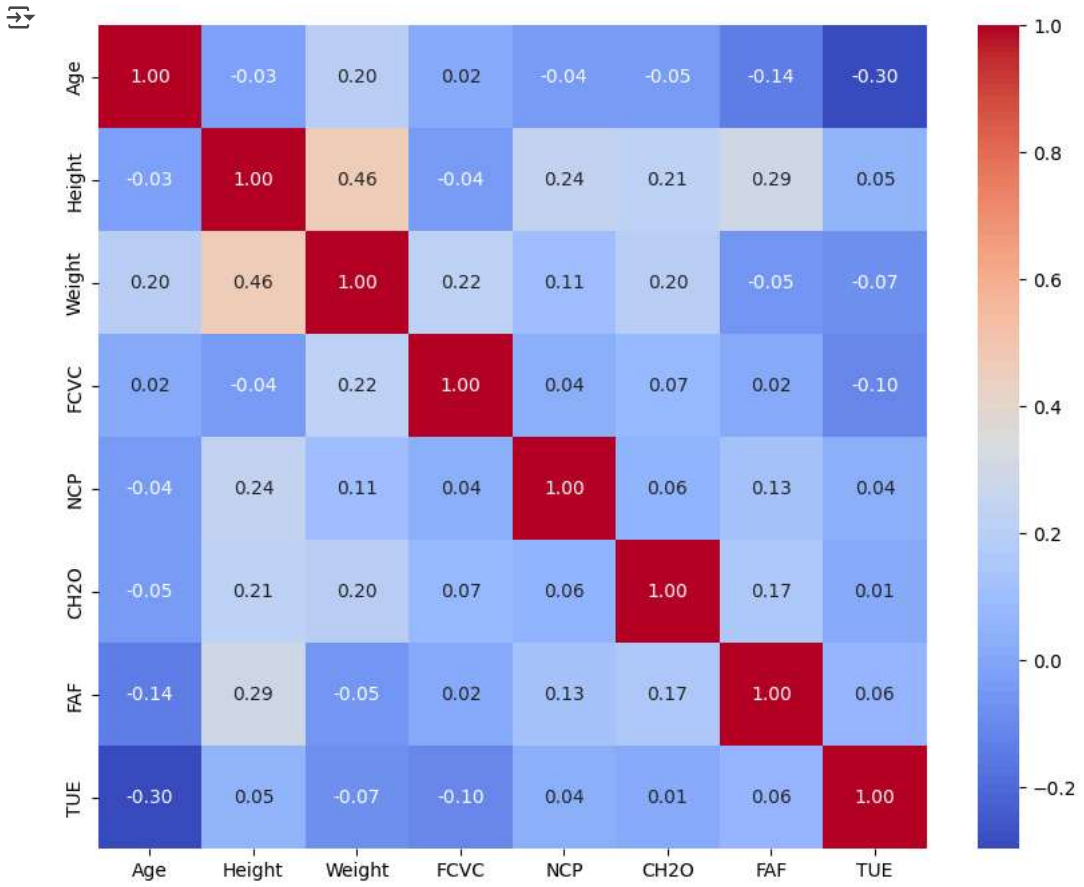
Instructions:

1. Compute the correlation matrix for the numerical features in the Obesity dataset.
2. Visualize this correlation matrix using a heatmap to make it easier to identify relationships between features.
3. Analyze the heatmap to pinpoint any notable feature correlations. Pay particular attention to features with high correlation coefficients, as these relationships are particularly impactful.

Question:

Based on your correlation matrix analysis for the Obesity dataset, identify one pair of features that exhibit a high degree of correlation. Describe how this finding could influence the feature selection process for a kNN model. What might be the implications of including both of these highly correlated features in your kNN model?

```
import seaborn as sns

# Compute the correlation matrix for numerical features
correlation_matrix = df.select_dtypes(include=[np.number]).corr()  # Fill in the blank to compute correlation

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')  # Fill in the blank to plot the heatmap

# Show the plot
plt.show()
```



Height and Weight shows noticeble correlation

Covariance vs Correlation

Context:

In the field of statistics and data science, grasping the differences between covariance and correlation is crucial for analyzing relationships between two variables. Both metrics serve to indicate how variables change together, yet they offer distinct perspectives and interpretative values.

Question:

Which of the following statements correctly defines covariance and correlation?

Statements to Evaluate:

S1: Covariance measures how two variables change together, but its value is influenced by the scale of the variables, making it hard to interpret the strength of their relationship.

S2: Correlation is a normalized measure of covariance, providing a dimensionless value that indicates the strength and direction of a linear relationship.

S3: Correlation measures both linear and non-linear relationships between variables.

S4: Covariance and correlation both provide identical information about the direction and strength of the relationship between variables.

Options:

A) S1 and S2 : **Correct**

B) S1 and S3

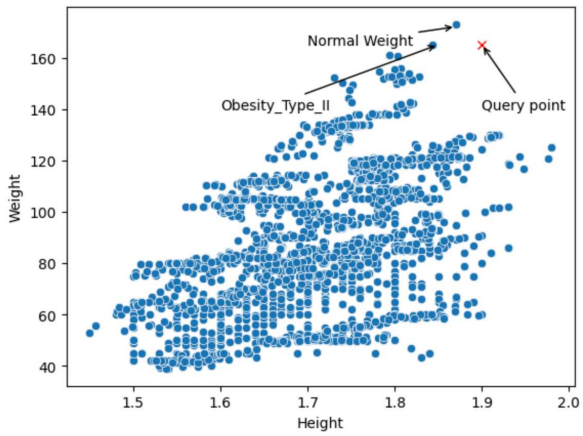
C) S2 and S4

D) S3 and S4

Select NearestNeighbor

Question

Suppose you are using kNN with k = 1 and have 2 features: height and weight. Then what will be the class for the query point ?



Options

A) Normal Weight : **Correct**

B) Obesity_Type_II

C) Either Normal Weight or Obesity_Type_II

D) Cannot be Determined

Note: **Select the correct option in the question link**

Know about kNN

Question

As kNN is a Non-parametric Algorithm, when is it helpful ?

Options

A) Nonparametric methods are commonly used when we want a few assumptions in our data: **Correct**

B) Nonparametric methods are commonly used when we want a lot of assumptions in our data

C) Nonparametric methods are commonly used when we want outliers in data

D) None of the above

Note: **Select the correct option in the question link**

Know your data

Context:

Effective data analysis often requires identifying which features in a dataset contain the most outliers and which have the largest range of values. This insight is crucial for optimizing machine learning algorithms, particularly those like kNN, which are sensitive to outliers and the scale of data.

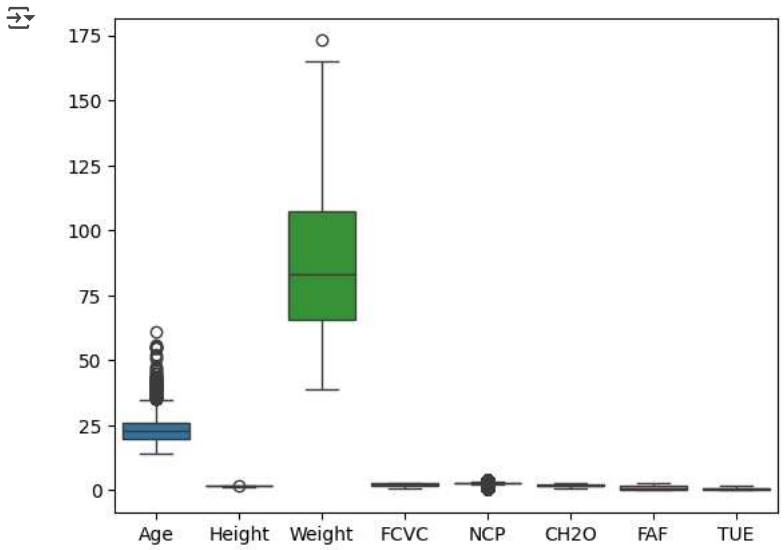
Task:

Determine which feature in our dataset has:

1. The highest number of outliers and
2. The largest range of values.

These characteristics are key to understanding data distribution and preparing for accurate modeling.

```
sns.boxplot(data = df) # Select plot that is useful in visualizing the distribution
plt.show()
```



The highest number of outliers:Age

The largest range of values:Weight

Euclidean Distance code

Context:

Understanding the concept of Euclidean distance is essential in machine learning, particularly in algorithms like kNN. This distance measure plays a crucial role in determining the similarity between data points.

Task:

Write Python code to standardize a dataset and calculate the Euclidean distance between the first and last data points using specific features. This exercise will help you grasp the practical application of Euclidean distance in data preprocessing and similarity measurement.

Instructions:

1. Select the subset of the entire dataset with only two features 'FCVC' and 'FAF.
2. Use `StandardScaler` to standardize this subset dataset.
3. Define a function to calculate the Euclidean distance between two data points using the 'FCVC' and 'FAF' features.
4. Calculate the Euclidean distance between the first and last data points in the entire scaled dataset.

Question:

After standardizing the dataset and calculating the Euclidean distance between the first and last data points using 'FCVC' and 'FAF' features, within what range does the calculated distance fall?

```
from sklearn.preprocessing import StandardScaler
from math import sqrt

def calculate_euclidean_distance(data_point1, data_point2):
    distance = 0.0
    distance = sqrt((data_point1['FCVC'] - data_point2['FCVC'])**2 + (distance))  # Try to remember the formula for euclidean distance
    return distance

df_scaler = StandardScaler()
scaled_data = df_scaler.fit_transform(df[['FCVC', 'FAF']])
scaled_data_df = pd.DataFrame(scaled_data, columns = ['FCVC', 'FAF'])

# Select the first and last data point
first_data_point = scaled_data_df.iloc[0]
last_data_point = scaled_data_df.iloc[-1]

# Calculate the Euclidean distance between the first and last data points
```

```
distance = calculate_euclidean_distance(first_data_point, last_data_point)
print(distance)
```

```
1.8733604563356185
```

Implementing kNN code

Context:

You're provided with a Python script implementing the k-Nearest Neighbors (kNN) algorithm from scratch. This script includes functions for calculating Euclidean distance and predicting the label of a new data point based on a set of labeled data. You'll fill in the missing parts of the code to make it functional.

Task:

Complete the provided Python code by filling in the blanks to enable the kNN algorithm to predict the label of a new data point based on 'Gender', 'Age', 'Height', and 'Weight'.

Question:

After completing and executing the code , what label does the kNN algorithm predict?

Fun Activity:

Try inputting your age, gender, height, and weight into the new data point and run the code to see what label the kNN algorithm predicts for you.

```
original_df = df.copy()


import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from math import sqrt
from collections import Counter

# Function to calculate Euclidean distance
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)):
        distance += (row1.iloc[i] - row2.iloc[i]) ** 2
    return sqrt(distance)  # Hint: Final distance formula calculation step



# Function to predict the label of a new data point
def predict_label(train_data, new_data_point, num_neighbors):
    distances = []
    for index, row in train_data.iterrows():
        dist = euclidean_distance(new_data_point, row[:-1])  # Hint: Calculate distance
        distances.append((row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = [dist[0] for dist in distances[:num_neighbors]]  # Hint: Neighbors selection

    output_values = [row.iloc[-1] for row in neighbors]
    prediction = Counter(output_values).most_common(1)[0][0]  # Hint: Determine majority using Counter

    return prediction



# Encoding the 'Gender' column
label_encoder = LabelEncoder()
original_df['Gender'] = label_encoder.fit_transform(original_df['Gender'])  # Hint: Fit and apply encoding

# Scaling the features
scaler = StandardScaler()
original_df[['Age', 'Height', 'Weight']] = scaler.fit_transform(original_df[['Age', 'Height', 'Weight']])

# New data point
new_data = {'Gender': 'Male', 'Age': 32, 'Height': 175, 'Weight': 75}
new_data_df = pd.DataFrame([new_data])

# Encoding and scaling the new data point
new_data_label_encoder = LabelEncoder() # Create a new LabelEncoder for the new data point
```

```
new_data_label_encoder = LabelEncoder() # Create a new LabelEncoder for the new data point
new_data_df['Gender'] = new_data_label_encoder.fit_transform(new_data_df['Gender']) # Fit and transform the new data point's gender
new_data_scaled = scaler.transform(new_data_df[['Age', 'Height', 'Weight']])  # Hint: Apply existing scaling

new_data_point = pd.Series([new_data_df['Gender'].iloc[0]] + new_data_scaled.tolist()[0])


# Predicting the label
predicted_label = predict_label(original_df, new_data_point, 5)
print(f"The predicted label is: {predicted_label}")
```

```
The predicted label is: Obesity_Type_I
```

Start coding or generate with AI.

Start coding or generate with AI.

KNN Lecture-2 Assesments

Scaling in kNN

Question:

Select the statements which are correct regarding the importance of scaling in kNN:

Statements:

S1: Scaling is important in kNN because it ensures that all features contribute equally to the distance computation.

S2: Unscaled features can disproportionately influence the distance calculations, affecting the performance of kNN.

S3: Scaling prevents features from having negative values.

S4: Scaling decreases the sample size of our data.

Options

A] S1 and S2 **CORRECT**

B] S1, S2, and S3

C] S2 and S4

D] S1, S2, and S4

Note: **Select the correct option in the question link**

Predicting kNN Model Accuracy

Context:

You're tasked with implementing a k-Nearest Neighbors (kNN) classifier to predict a target variable in a dataset. Before diving into the code, it's crucial to understand the preprocessing steps involved, such as label encoding and feature scaling, which can significantly impact the model's performance.

Objective:

Write a Python script to preprocess the data, train a kNN model, and predict its test accuracy. You will use label encoding for categorical features and standard scaling for numerical features before training the model.

Instructions:

1. **Label Encoding:**
    - Apply label encoding to convert all categorical features into numeric values. Use `LabelEncoder` from `sklearn.preprocessing` for this task.
2. **Feature and Target Selection:**
    - Separate your features (`X`) and target variable (`y`).

3. **Data Splitting:**
    - Split the data into training and testing sets using `train_test_split` from `sklearn.model_selection`. Set `test_size` to 0.33 and `random_state` to 42.
4. **Feature Scaling:**
    - Standardize the features using `StandardScaler` from `sklearn.preprocessing`. Fit the scaler on the training set and transform both the training and testing sets.
5. **Model Training:**
    - Initialize and train a `KNeighborsClassifier` from `sklearn.neighbors` with default parameters. Fit the model on the scaled training data.
6. **Model Evaluation:**
    - Evaluate the model's accuracy on the scaled test set using the `score` method and print the result.

Question:

After writing and executing the code, within what range does the kNN model's test accuracy fall?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier

# Apply label encoding
label_encoder = LabelEncoder()
for column in df.select_dtypes("object").columns: # Hint: Select only object columns
    df[column] = label_encoder.fit_transform(df[column]) # Hint: fit and transform the categorical column.

X = df.iloc[:, :-1]  # Features
y = df.iloc[:, -1]   # Target variable

# Split the data using train_test_split function
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train) # Hint: fit and transform the X_train data.
X_test_scaled = scaler.transform(X_test) # Hint: only transform the X_test data.

# Train the kNN model
knn = KNeighborsClassifier()
knn.fit(X_train_scaled,y_train) # Hint: Fit the model using X scaled and y_train.

# Print the accuracy
print(knn.score(X_test_scaled,y_test)) # Hint: check the accuracy on test dataset.
```

```
0.8005738880918221
```

---

kNN - Distance Metrics

Context:

In this exercise, you will explore the impact of different distance metrics on the k-Nearest Neighbors (kNN) model's performance. By comparing the test accuracy of the kNN model using various distance metrics, you can identify which metric is most effective for this particular dataset.

Task:

Implement a Python script to train and evaluate kNN models using different distance metrics. Your goal is to determine which distance metric leads to the highest test accuracy.

Instructions:

1. **Distance Metrics:** Define a list of distance metrics to evaluate – Euclidean, Manhattan, and Cosine.
2. **Model Training and Evaluation:** For each distance metric, train a kNN model on the scaled training data and evaluate its accuracy on the scaled test data.
3. **Accuracy Comparison:** Store and compare the accuracy scores for each metric to identify the most effective one.

Question:

Based on the script, which distance metric provides the best test accuracy for the kNN model?

```
from sklearn.metrics import accuracy_score # Hint: Import accuracy_score

# Defining different distance metrics
metrics = ['euclidean', 'manhattan', 'cosine']
scores = {}

# Training and evaluating a kNN model for each distance metric

for metric in metrics: # Hint: iterate over the metrics list
    knn = KNeighborsClassifier(metric=metric) # Hint: Use metric hyperparam.
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled) # Hint: make the prediction on test data.
    score = accuracy_score(y_test,y_pred) # Hint: check the accuracy on the test data predictions.
    scores[metric] = score
    print(f"Accuracy with {metric} distance: {score}")

# Determining the best performing metric
best_metric = max(scores, key=scores.get)
print(f"The best-performing metric is {best_metric} with an accuracy of {scores[best_metric]}.")
```

```
Accuracy with euclidean distance: 0.8005738880918221
Accuracy with manhattan distance: 0.8622668579626973
Accuracy with cosine distance: 0.8020086083213773
The best-performing metric is manhattan with an accuracy of 0.8622668579626973.
```

Bias-Variance in kNN

Context:

To understand the bias-variance trade-off in the k-Nearest Neighbors (kNN) algorithm, you'll analyze how kNN's accuracy varies with different k values. By plotting the training and testing accuracies on the same graph, you can visually assess how the model's bias and variance change as the complexity (determined by k) changes.

Task:

Implement Python code to train kNN models with a range of k values and plot the training and testing accuracies on a multi-line graph. This visualization will help you determine the optimal balance between bias and variance.

Instructions:

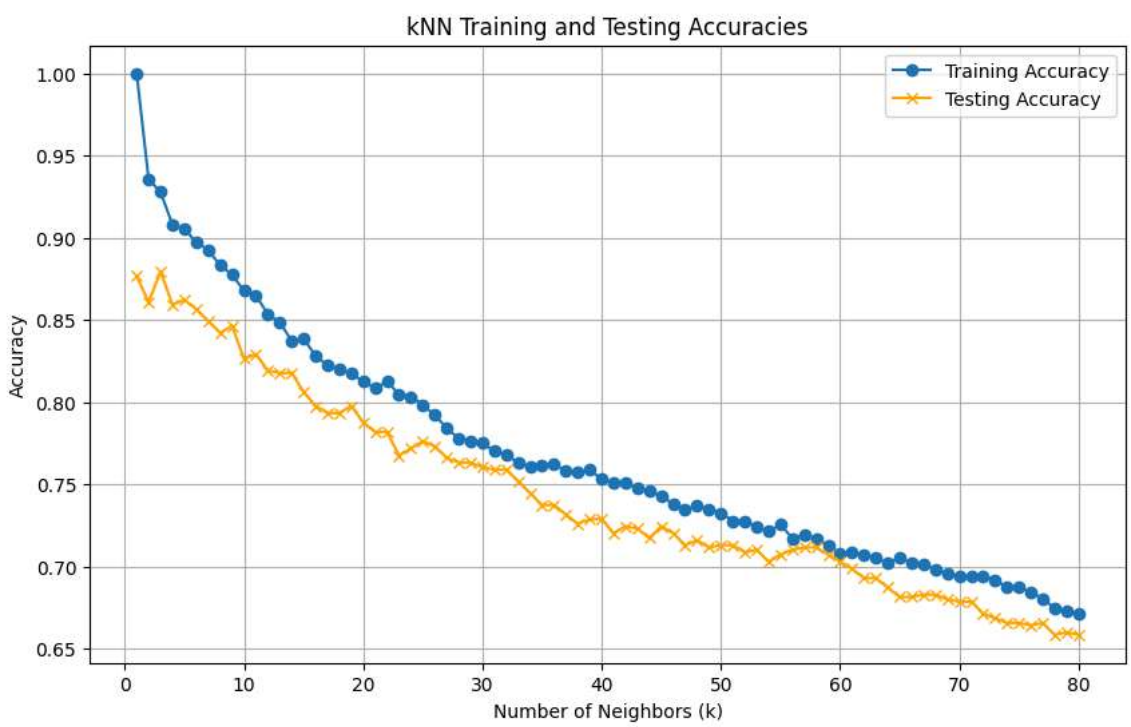
1. **Initialize Accuracy Lists:** Create two lists, `train_acc` and `test_acc`, to store the accuracies for training and testing sets, respectively.
2. **Training and Evaluation Loop:** Iterate k from 1 to 80, train a kNN model using the Manhattan metric for each k using the previously used scaled data, and append the accuracies to the corresponding lists.
3. **Plotting:** Use `matplotlib` to create a multi-line plot where the x-axis represents the k values, and the y-axis represents accuracy. Plot both `train_acc` and `test_acc` on this graph to create two distinct lines.
4. **Enhancing the Plot:**
    - Add a title to the plot: 'kNN Training and Testing Accuracies'.
    - Label the x-axis as 'Number of Neighbors (k)' and the y-axis as 'Accuracy'.
    - Insert a legend to differentiate between the training and testing accuracy lines.
    - Use different colors for the two lines for clear visualization.

Question:

After visualizing the training and testing accuracies for different k values, how do the trends in the plot help you understand the bias-variance trade-off in kNN?

```
train_acc = [] # Hint: empty list
test_acc = [] # Hint: empty list

for k in range(1, 81): # Hint: iterate over 1 to 80.
    kNN = KNeighborsClassifier(n_neighbors=k, metric="manhattan") # Hint: put appropriate hyperparams.
    kNN.fit(X_train_scaled, y_train)
    train_acc.append(kNN.score(X_train_scaled, y_train)) # Hint: score on Train dataset.
    test_acc.append(kNN.score(X_test_scaled, y_test)) # Hint: score on Test dataset.
```

```
plt.figure(figsize=(10, 6))
plt.plot(range(1, 81), train_acc, label='Training Accuracy', marker='o') # Hint: Plot train accuracies.
plt.plot(range(1, 81), test_acc, color='orange', label='Testing Accuracy', marker='x') # Hint: Plot train accuracies.
plt.title('kNN Training and Testing Accuracies')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



Initially, bias is low and variance is high; later, bias increases and variance decreases.

---

kNN Properties

Context:

The k-Nearest Neighbors (kNN) algorithm is widely used in machine learning for its simplicity and effectiveness. However, it has certain characteristics and challenges, especially related to dimensionality, distance measures, noise, and the choice of k.

Question:

Which of the following statements is/are correct regarding the kNN algorithm?

Statements:

S1: As the number of features increases, the requirement for data points increases exponentially.

S2: With increasing dimensions, the Euclidean distance loses its significance.

S3: kNN is sensitive to noise and has high time complexity, particularly because it does not learn a discriminative function from the training data but uses the entire dataset for prediction.

S4: A lower value of k in kNN can lead to overfitting, as the predictions become more sensitive to the noise in the training data.

Options

A] S1, S2, and S3

B] S1 and S2

C] S1, S3, and S4

D] All of the Statements

Note: **Select the correct option in the question link**

kNN Properties-Continued

Context:

The k-Nearest Neighbors (kNN) algorithm has unique properties and approaches that address its computational challenges and the peculiarities of its application. Understanding these properties can greatly enhance the effectiveness of kNN in practical scenarios.

Question:

Which of the following statements are correct regarding the properties and characteristics of the kNN algorithm?

Statements:

S1: If our data is large, locality-sensitive hashing (LSH) can be used to map data to a virtual grid, accelerating the kNN search process.

S2: kNN can easily handle outlier data by choosing a very high value of k, as it averages the neighbors' responses, potentially diminishing the effect of outliers.

S3: kNN is considered a non-parametric method because it does not assume a particular form for the function that describes the relationship between feature and target variables.

Options

A] S1 and S3 **Correct**

B] S2 only

C] S1 only

D] S1, S2, and S3

Note: **Select the correct option in the question link** As the number of features increases, the requirement for data points increases exponentially. because with greater no. of feature sparsity increases. In a high-dimensional space, points tend to be farther apart. Imagine a cube: with 3D, it's easy to fill, but in 100D, most of the 'cube' is empty. This is why we need more data points as dimensions increase. Does this clarify the concept?

Optimizing kNN with Cross-Validation

Context:

You're tasked with writing Python code to determine the optimal k value for a k-Nearest Neighbors (kNN) classifier using cross-validation. This method will help you understand how different k values influence the model's predictive performance.

Task:

Write a Python script to find the optimal k value for a kNN classifier by testing a range of k values and evaluating their performance using cross-validation. Then, assess the chosen k value's performance on a test set.

Instructions:

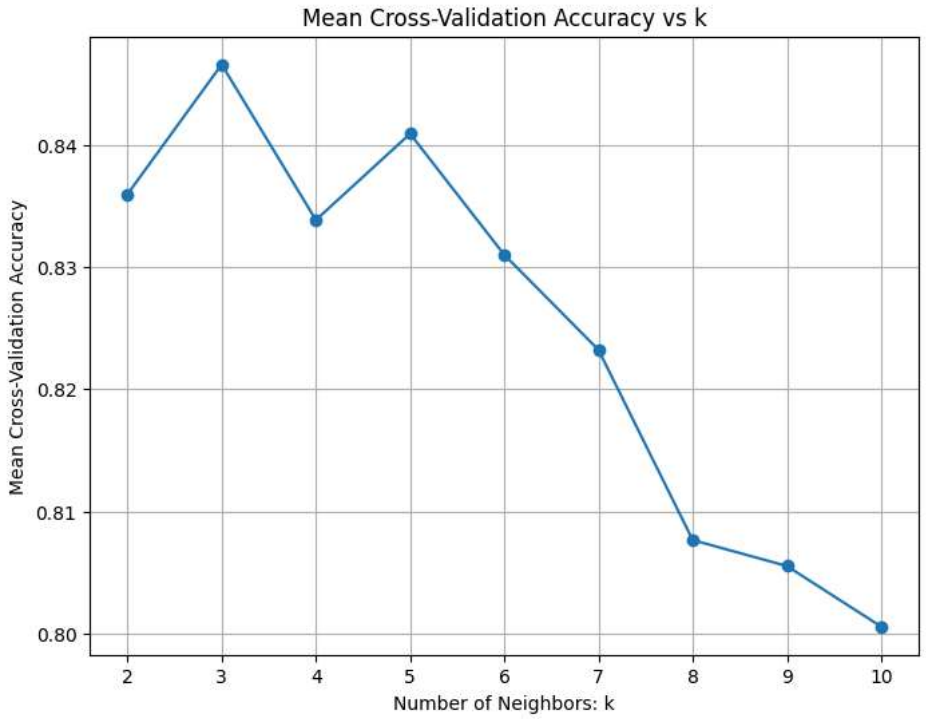
1. **Define the Range for k:** Set up a range of k values from 2 to 10.
2. **Cross-Validation Loop:** Iterate through the k values, initializing a KNeighborsClassifier for each. Perform 5-fold cross-validation on the scaled training data and compute the mean accuracy for each k.
3. **Identify the Best k:** Determine which k value yields the highest mean cross-validation accuracy.
4. **Test Set Evaluation:** Train a new kNN model using the best k value on the entire training set. Then, evaluate this model on the test set to obtain the test accuracy.
5. **Code Execution:** Execute your script to discover the optimal k value and its corresponding test set accuracy.

Question:

Based on your script execution, which statement correctly describes the outcome for the optimal k value and its test set accuracy?

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

# Array to store mean cross-validation scores
```

```
mean_cv_scores = []

# Define the range of k values to try
k_values = np.arange(2, 11)


for k in k_values: # Hint: # Loop over each value of k

    knn = KNeighborsClassifier(n_neighbors=k, metric='manhattan') # Hint: Initialize the kNN classifier with the current k value.

    cv_scores = cross_val_score(knn, X_train_scaled, y_train, cv=5) # Hint: Perform 5-fold cross-validation.

    mean_cv_scores.append(np.mean(cv_scores)) # Hint: Calculate the mean of the cross-validation scores.


plt.figure(figsize=(8, 6))
plt.plot(k_values, mean_cv_scores, marker='o') # Hint: Plotting the mean cross-validation scores for each k value
plt.title('Mean Cross-Validation Accuracy vs k')
plt.xlabel('Number of Neighbors: k')
plt.ylabel('Mean Cross-Validation Accuracy')
plt.grid(True)
plt.show()
```




```
best_k = k_values[np.argmax(mean_cv_scores)] # Hint:  Identify the k value with the highest mean cross-validation accuracy from the plot

print(f"The best value of k is: {best_k} with a cross-validation accuracy of: {max(mean_cv_scores):.2f}")


knn_best = KNeighborsClassifier(n_neighbors=best_k, metric='manhattan')
knn_best.fit(X_train_scaled, y_train)  # Hint: Training the model on train dataset with the best k value
test_accuracy = knn_best.score(X_test_scaled, y_test) # Hint: evaluate model performance on the test set
print(f"Test set accuracy with the best k: {test_accuracy}")
```

```
The best value of k is: 3 with a cross-validation accuracy of: 0.85
Test set accuracy with the best k: 0.8794835007173601
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.