Certainly! Let's structure the information about the k-Nearest Neighbors (KNN) algorithm and the Decision Tree algorithm into the requested sections.

k-Nearest Neighbors (KNN):

Title: Understanding k-Nearest Neighbors (KNN) Algorithm

Introduction: K-Nearest Neighbors (KNN) is an instance-based learning algorithm used for both classification and regression tasks. Rather than relying on a predefined model, KNN makes predictions based on the similarity between new instances and instances in the training dataset. It is a non-parametric and lazy learning algorithm, meaning it doesn't have a distinct training phase, and predictions are made at runtime based on the similarity between instances.

Method/Working:

1. **Instance-Based Learning:**
   - KNN is an instance-based learning algorithm where predictions are made based on the majority class or average of the k nearest neighbors of a new instance.
   - The algorithm classifies a new instance by a majority vote of its k nearest neighbors in the feature space.
2. **Lazy Learning:**
   - KNN doesn't construct an explicit model during the training phase. Instead, it memorizes the training dataset and performs computations at the time of prediction.
   - The choice of the parameter 'k' (number of neighbors) is crucial and impacts the model's performance.

Decision Tree:

Title: Exploring the Decision Tree Algorithm

Introduction: Decision Tree is a versatile and widely used machine learning algorithm that belongs to the family of tree-based models. It is employed for both classification and regression tasks, offering a clear and interpretable decision-making structure. The algorithm recursively splits the dataset into subsets based on the most significant attribute at each node, creating a tree-like structure where each internal node represents a decision based on an attribute, and each leaf node represents the outcome.

Method/Working:

1. **Tree-Based Learning:**
   - Decision Tree builds a tree-like structure where each node represents a decision based on a specific attribute.
   - The tree is constructed based on recursive splitting, with each internal node representing a decision, and each leaf node representing the final outcome.
2. **Eager Learning:**
   - Decision Tree is an eager learning algorithm, meaning it constructs the entire tree during the training phase.
   - The construction involves selecting the best attribute to split the dataset at each node, optimizing a predefined criterion (e.g., Gini impurity for classification or mean squared error for regression).

These sections provide an introduction to the k-Nearest Neighbors (KNN) and Decision Tree algorithms, along with an overview of their working principles. If you have any further questions or if there's anything specific you'd like to discuss, feel free to let me know!

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
%matplotlib inline
print("Imported!")
```

```
Imported!
```

```
# collectiong data
file = pd.read_csv("/content/Gender_classification_dataset.csv")
file.isna().sum()
```

```
long_hair                   0
forehead_width_cm           0
forehead_height_cm          0
nose_wide                   0
nose_long                   0
lips_thin                   0
distance_nose_to_lip_long   0
gender                      0
dtype: int64
```

```
file.head(15)
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin
0	1	11.8	6.1	1	0	1
1	0	14.0	5.4	0	0	1
2	0	11.8	6.3	1	1	1
3	0	14.4	6.1	0	1	1
4	1	13.5	5.9	0	0	0
5	1	13.0	6.8	1	1	1
6	1	15.3	6.2	1	1	1
7	0	13.0	5.2	0	0	0
8	1	11.9	5.4	1	0	1
9	1	12.1	5.4	0	0	0
10	0	12.5	5.4	1	1	1
11	1	15.5	5.8	1	1	1
12	0	14.7	5.2	1	1	1
13	1	14.5	6.7	0	1	1
14	1	14.2	6.5	0	0	0

```
print(file.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5001 entries, 0 to 5000
Data columns (total 8 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   long_hair                  5001 non-null   int64
 1   forehead_width_cm          5001 non-null   float64
 2   forehead_height_cm         5001 non-null   float64
 3   nose_wide                  5001 non-null   int64
 4   nose_long                  5001 non-null   int64
 5   lips_thin                  5001 non-null   int64
 6   distance_nose_to_lip_long  5001 non-null   int64
 7   gender                     5001 non-null   int64
dtypes: float64(2), int64(6)
memory usage: 312.7 KB
None
```

```
x = file.drop("gender",axis = 1)
y = file["gender"]

x
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_th
0	1	11.8	6.1	1	0	
1	0	14.0	5.4	0	0	
2	0	11.8	6.3	1	1	
3	0	14.4	6.1	0	1	
4	1	13.5	5.9	0	0	
...	...	...	...	...	...	
4996	1	13.6	5.1	0	0	
4997	1	11.9	5.4	0	0	
4998	1	12.9	5.7	0	0	
4999	1	13.2	6.2	0	0	
5000	1	15.4	5.4	1	1	

5001 rows × 7 columns

```
# fillna on x or features

x = x.fillna(x.mean())
x.isna().sum()
```

```
long_hair                   0
forehead_width_cm           0
forehead_height_cm          0
nose_wide                   0
nose_long                   0
lips_thin                   0
distance_nose_to_lip_long   0
dtype: int64
```

```
from sklearn.model_selection import train_test_split

np.random.seed(3)

x_train,x_test,y_train,y_test  = train_test_split(x,y,test_size=0.2)

x_train
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_th
2277	1	12.2	5.3	1	1	
3924	0	13.3	6.9	1	1	
2212	1	14.8	5.7	1	1	
1081	1	11.9	6.1	1	1	
4769	1	14.9	5.3	1	1	
...	...	...	...	...	...	
789	1	13.2	7.0	1	0	
968	1	14.2	6.2	0	1	
1667	1	14.0	5.2	1	1	
3321	1	12.9	5.5	1	1	
1688	1	13.5	5.4	0	1	

4000 rows × 7 columns

```
from sklearn import tree

model1 = tree.DecisionTreeClassifier()

model1.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
print(f"The accuracy of the Decision tree classifier model is :{model1.score(x_test,y_test)*100}%")
```

```
The accuracy of the Decision tree classifier model is :96.30369630369631%
```

```
from sklearn import neighbors
model2 = neighbors.KNeighborsClassifier()

model2.fit(x_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
print(f"The accuracy of the K nearest Neighbour classifier model is :{model2.score(x_test,y_test)*100}%")
```

```
The accuracy of the K nearest Neighbour classifier model is :97.1028971028971%
```