Naïve Bayes Exercise 1: Applying the Naïve Bayes Algorithm

Now, let's apply the Naïve Bayes algorithm to a Fertility dataset, which aims to determine whether the fertility level of an individual has been affected by their demographics, their environmental conditions, and their previous medical conditions. Follow these steps to complete this exercise:

1. Download the Fertility dataset from http://archive.ics.uci.edu/ml/datasets/Fertility. Go to the link and click on **`Data Folder`**. Click on **`fertility_Diagnosis.txt`**, which will trigger the download. Save it as a **`.csv`** file.
2. Open a Jupyter Notebook to implement this exercise. Import pandas, as well as the **`GaussianNB`** class from scikit-learn's **`naive_bayes`** module:
   ```
   import pandas as pd
   from sklearn.naive_bayes import
   GaussianNB
   ```
3. Read the **`.csv`** file that you downloaded in the first step. Make sure that you add the **`header`** argument equal to **`None`** to the **`read_csv`** function, considering that the dataset does not contain a header row:

```
data =
pd.read_csv("fertility_Diagnosis.csv",
header=None)
```

4. Split the data into **X** and **Y**, considering that the class label is found under the column with an index equal to 9. Use the following code to do so:

```
X = data.iloc[:,:9]
Y = data.iloc[:,9]
```

5. Instantiate the **GaussianNB** class that we imported previously. Next, use the **fit** method to train the model using **X** and **Y**:

```
model = GaussianNB()
model.fit(X, Y)
```

The output from running this script is as follows:

```
GaussianNB(priors=None,
var_smoothing=1e-09)
```

This states that the instantiation of the class was successful. The information inside the parentheses represents the values used for the arguments that the class accepts, which are the hyperparameters.

For instance, for the **GaussianNB** class, it is possible to set the prior probabilities to consider for each class label and a smoothing argument that stabilizes variance. Nonetheless, the model was initialized without setting any arguments, which means that it will use the default values for each argument, which is **`None`** for the case of **`priors`** and **`1e-09`** for the smoothing hyperparameter.

6. Finally, perform a prediction using the model that you trained before, for a new instance with the following values for each feature: **`−0.33`**, **`0.69`**, **`0`**, **`1`**, **`1`**, **`0`**, **`0.8`**, **`0`**, **`0.88`**. Use the following code to do so:

```
pred = model.predict([[-
0.33,0.69,0,1,1,0,0.8,0,0.88]])
print(pred)
```

Note that we feed the values inside of double square brackets, considering that the **`predict`** function takes in the values for prediction as an array of arrays, where the first set of arrays corresponds to the list of new instances to predict and the second array refers to the list of features for each instance.

The output from the preceding code snippet is as follows:

```
['N']
```

The predicted class for that subject is equal to **`N`**, which means that the fertility of the subject has not been affected.

You have successfully trained a Naïve Bayes model and performed prediction on a new observation.

SVM Exercise 2: Applying the SVM Algorithm

In this exercise, we will apply the SVM algorithm to the Fertility dataset. The idea, which is the same as in previous exercise, is to determine whether the fertility level of an individual is affected by their demographics, their environmental conditions, and their previous medical conditions. Follow these steps to complete this exercise:

1. Open a Jupyter Notebook to implement this exercise. Import pandas as well as the **`SVC`** class from scikit-learn's **`svm`** module:
   ```
   import pandas as pd
   from sklearn.svm import SVC
   ```
2. Load the **`fertility_Diagnosis`** dataset that you downloaded in *Exercise 1*, *Applying the Naïve Bayes Algorithm*. Make sure to add the **`header = None`** argument to the **`read_csv`** function, considering that the dataset does not contain a header row:
   ```
   data =
   pd.read_csv("fertility_Diagnosis.csv",
   header=None)
   ```

3. Split the data into **X** and **Y**, considering that the class label is found under the column with the index equal to **9**. Use the following code to do so:
   ```
   X = data.iloc[:,:9]
   Y = data.iloc[:,9]
   ```
4. Instantiate scikit-learn's **SVC** class and use the **fit** function to train the model using **X** and **Y**:
   ```
   model = SVC()
   model.fit(X, Y)
   ```

Again, the output from running this code represents the summary of the model, along with its default hyperparameters, as follows:

```
SVC(C=1.0, break_ties=False,
cache_size=200,
    class_weight=None, coef0=0.0,
    decision_function_shape='ovr',
degree=3,
    gamma='scale', kernel='rbf',
max_iter=-1,
    probability=False,
random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

5. Finally, perform a prediction using the model that you trained previously, for the same instances that we used in *Exercise 4.01*, *Applying the Naïve Bayes Algorithm*: **-0.33**, **0.69**, **0**, **1**, **1**, **0**, **0.8**, **0**, **0.88**. Use the following code to do so:

```
pred = model.predict([[-
0.33,0.69,0,1,1,0,0.8,0,0.88]])
print(pred)
```

The output is as follows:

```
['N']
```

Again, the model predicts the instance's class label as **N**, meaning that the fertility of the subject has not been affected.

You have successfully trained an SVM model and performed a prediction.