that are already in the data set: 1 for North America, 2 for Europe, and 3 for Asia are not a good a representation, because this imposes an ordering on the data. There is no particular reason why values should be ordered in this manner and results could end up being biased. We fix this be introducing a new encoding where each value that a category can take on is represented by a different feature. Since **'origin'** can take on three values, we add three new features, i.e., three more columns to the feature matrix. These columns could be called **'Amer'**, **'Euro'** and **'Asia'**. The values in the columns will be set to either **1** or **0**, according to the following rule:

- If **'origin'=1** set **'Amer'=1**, **'Euro'=0**, and **'Asia'=0**
- If **'origin'=2** set **'Amer'=0**, **'Euro'=1**, and **'Asia'=0**
- If **'origin'=3** set **'Amer'=0**, **'Euro'=0**, and **'Asia'=1**

This creates an encoding where each column only has a **1** in the individual cars corresponding to a particular origin, and otherwise has a **0** for that feature. This is called a **1-Hot** or **One-Hot** encoding. Each car has a one, or a hot value, only in the column corresponding to the correct category. The word **hot** is used in the sense of an electrical circuit: when it is hot, it is turned on, and when it is not hot, it is turned off.[7]

The **pandas** function **get_dummies** implements **One-Hot** encodings.[8] To demonstrate it we'll create a temporary new **DataFrame** with **origin** as a column and print some of the data.

```
DF=data[["mpg","displ","hp","accel","origin"]]
DF[16:21]  # print lines 16 through 21
```

	mpg	displ	hp	accel	origin
16	18.0	199.0	97.0	15.5	1
17	21.0	200.0	85.0	16.0	1
18	27.0	97.0	88.0	14.5	3
19	26.0	97.0	46.0	20.5	2
20	25.0	110.0	87.0	17.5	2

We then tell **get_dummies** which columns represent categorical variables by giving it a list of the column names. Since we no longer need **DF** we'll overwrite it with the modified **DataFrame**.

```
DF=pd.get_dummies(DF,columns=["origin"])
DF[16:21]
```

[7] Some modelers prefer to use one fewer new variable then there are categories and set the value for the last category to all zero's. In this case, an Asian car would be identified by setting *both* **'Amer'=0** and **'Euro'=0**, without the need for the third variable **'Asia'**.

[8] See https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

```
        mpg     displ   hp      accel   origin_1 origin_2 origin_3
16      18.0    199.0   97.0    15.5    1        0        0
17      21.0    200.0   85.0    16.0    1        0        0
18      27.0    97.0    88.0    14.5    0        0        1
19      26.0    97.0    46.0    20.5    0        1        0
20      25.0    110.0   87.0    17.5    0        1        0
```

Finally, convert the temporary data frame to a feature matrix.

```
X=np.array(DF)
print(X[16:21]) # print lines 16 through 20
```

```
[[ 18.  199.   97.   15.5   1.    0.    0. ]
 [ 21.  200.   85.   16.    1.    0.    0. ]
 [ 27.   97.   88.   14.5   0.    0.    1. ]
 [ 26.   97.   46.   20.5   0.    1.    0. ]
 [ 25.  110.   87.   17.5   0.    1.    0. ]]
```

Since we want to predict the number of cylinders as a category, starting with category zero, we'll use the **unis** list defined on the previous page:

```
Y=[unis.index(j) for j in Y]
```

This converts each 3 to a 0, 4 to a 1, 5 to a 2, 6 to 3, and 8 to a 4.

Here we run the Naive Bayes' method through 100 train/test splits.

```
errs=[]
for j in range(100):
    gnb=GaussianNB()
    XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
    gnb.fit(XTRAIN,YTRAIN)
    YP=gnb.predict(XTEST)
    errs.append(1-accuracy_score(YTEST,YP))
print("%d Splits: Mean Error=%7.6f +/- %7.6f (95%%)"\
      %(nsplits, np.mean(errs),1.96*np.std(errs)))
print("Last confusion matrix:")
print(confusion_matrix(YP,YTEST))
```

```
100 Splits: Mean Error=0.359490 +/- 0.147373 (95%)
Last confusion matrix:
[[ 0  0  0  0  0]
 [ 1 42  2  3  0]
 [ 0  0  0  0  0]
 [ 0  0  0  1  1]
 [ 0  0  0 22 26]]
```

Rather than improving our prediction accuracy, we appear to have gotten worse. The error has increased to over 35%.