Baseline models for classification

It's often useful to start off with a "naive" model for prediction.

This gives us a "baseline" against which we can compare the more sophisticated models that we endeavor to create.

What's a good baseline model for the Classification task ?

Here are some choices:

- Most Frequent: always predict the class that occurs most frequently
- Constant: always predict Positive
- Uniform: predict each class with equal probability (50% for binary case)
- Stratified: predict each class according to empirical distribution

Let's see the performance (Accuracy) of each model on the MNIST Binary Classification task

- Predict one particular digits (Positive)
- Versus the 9 other digits (Negative)

In [4]:

```
mnh_d = mnist_helper.MNIST_Helper()
mnh_d.setup()

# Turn the 10 class training set into a binary training set
# - Same examples, different targets
# - targets are now "is 'digit'" or "is not 'digit'" for a single digit
digit = '5'
y_train_d, y_test_d = mnh_d.make_binary(digit)
```

```
Retrieving MNIST_784 from cache
```

In [5]:

```
strats = { "stratified": {},
           "uniform": {},
           "most_frequent": {},
           "constant": {"constant": True}
         }

plt_num = 1

# Compute Accuracy for various baseline classifiers
for strat, args in strats.items():
    dmy_clf = DummyClassifier(strategy=strat, **args)
    acc_scores_dmy = cross_val_score(dmy_clf, mnh_d.X_train, y_train_d, cv=5, sc
oring="accuracy")

    print("{s}: Accuracy = {a:.2f}".format(s=strat, a=acc_scores_dmy.mean()))
```

```
stratified: Accuracy = 0.84
uniform: Accuracy = 0.50
most_frequent: Accuracy = 0.91
constant: Accuracy = 0.09
```

Note that we have a highly imbalanced dataset (only approximately 10% Positive examples)

This explains

- the poor accuracy of the Constant Baseline (should match the 10% of Positive examples)
- the good accuracy of the Most Frequent Baseline (should match the 90% of Negative examples)
- the good accuracy of the Stratified Baseline

Remember to take the imbalance into account when evaluating the Performance metric.

If we do a little fitting, we can come up with simple, non-trivial Baseline Models.

If our "complicated" model seems to improve on the simple baseline, then (perhaps) our efforts have achieved something.

The Naive Bayes model is a popular baseline.

In [6]:
```
print("Done")
```

```
Done
```