Machine Learning

| Multiclass Classification

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Multiclass Classification

Multiclass Classification

- Whereas **binary classifiers** distinguish between **two classes**, multiclass classifiers (also called **multinomial** classifiers) can distinguish between **more than** two classes.
- A classification task where each input sample should be categorized into more than two categories: for instance, classifying handwritten digits.
- Classifying news wires by topic
- A classification scheme looking at **movie descriptions** might try to classify them as "**action**," "**adventure**," "fantasy," "romance," "history" and the like.

Multiclass Classification

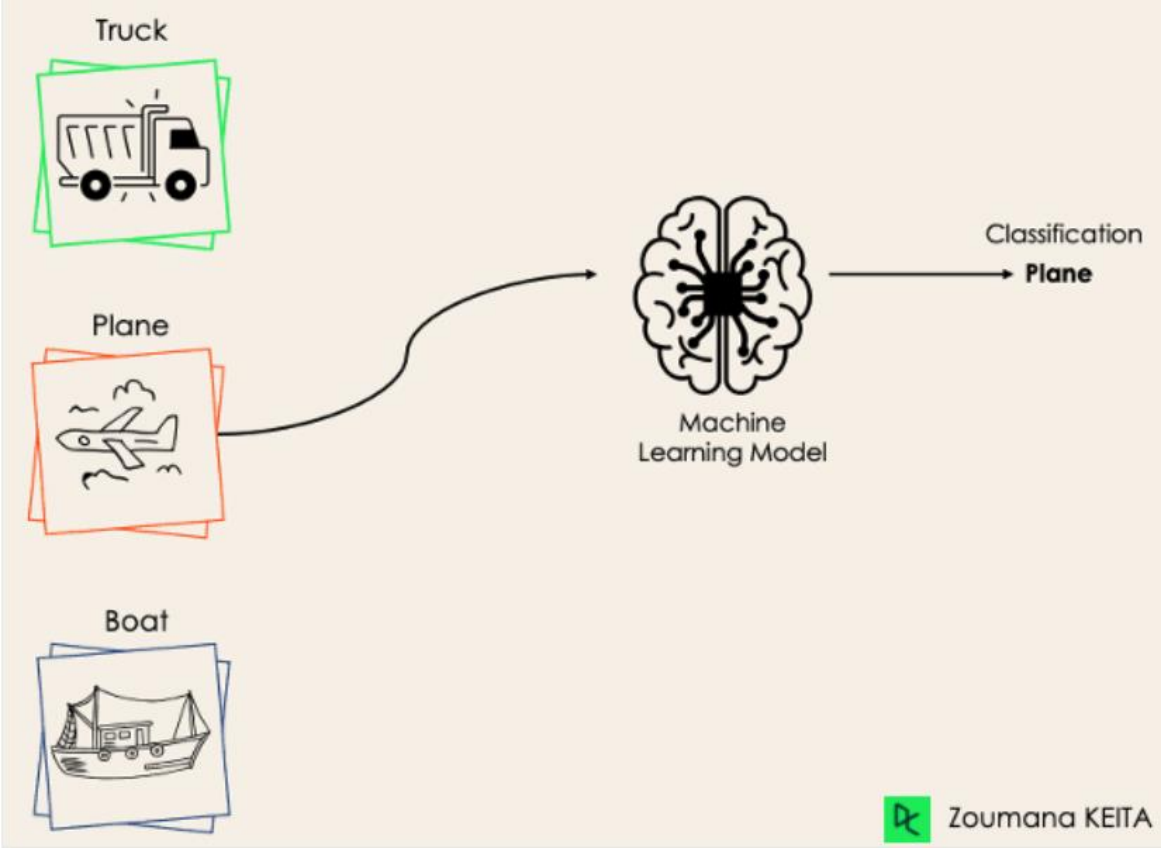
- In machine learning, a category in a classification problem is called a class.
- Data points are called samples.
- The class associated with a specific sample is called a label.

Multiclass Classification

- Multiclassification uses more than two classes, such as the **10 classes**, 0 through 9, in the **Digits** dataset.

Multiclass Classification

- The multi-class classification, on the other hand, has at least two mutually exclusive class labels, where the goal is to predict to which class a given input example belongs to.



Multiclass Classification

- Some algorithms are capable of handling multiple classes **directly**
  - LogisticRegression,
  - RandomForestClassifier, and
  - GaussianNB.
- Others are **strictly binary** classifiers (such as
  - SGDClassifier and
  - SVC).
- However, there are various strategies that you can use to perform multiclass classification using multiple binary classifiers
  - One-Versus-All OvA
  - One-Versus-One OvO

Multiclass Classification

- However, there are various strategies that you can use to perform multiclass classification using multiple binary classifiers
  - One-Versus-All OvA
  - One-Versus-One OvO

One-versus-all (OvA)

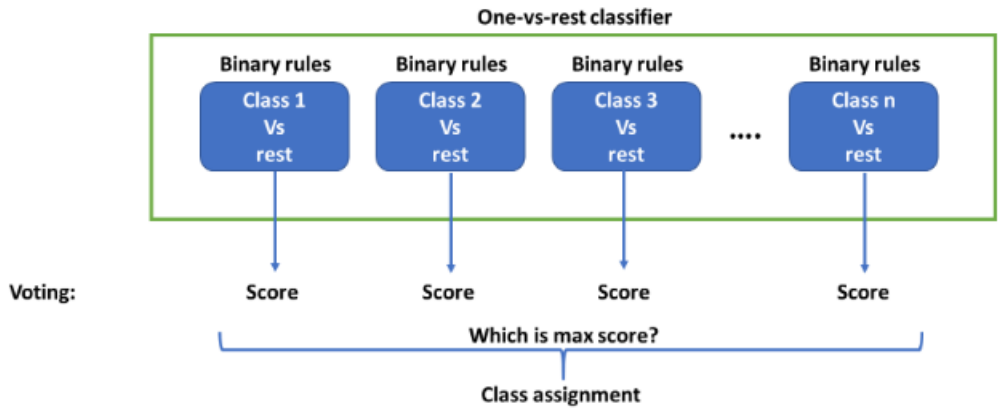
- OvA (also, one-versus-the-rest OvR) is a strategy used for multi-class classification problems. It's especially useful when your machine learning algorithm natively supports only binary classification
- For a classification problem with **N classes**, the OvA strategy involves training **N separate binary classifiers**. Each classifier is trained to distinguish between one of the classes and all the other classes combined.

One-versus-all (OvA)

- For instance, if you have three classes A, B, and C, you would train three classifiers:
  - Classifier 1 distinguishes A from B and C.
  - Classifier 2 distinguishes B from A and C.
  - Classifier 3 distinguishes C from A and B.

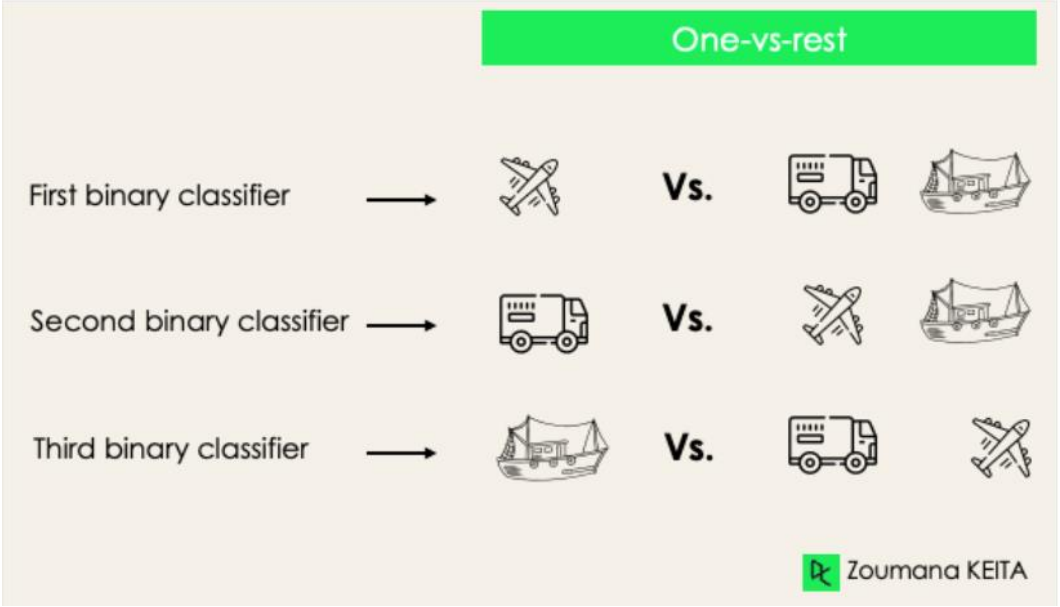
One-versus-all (OvA)

- **Training**: Each classifier is trained on all the data, where the examples of the target class are labeled as positive, and all other examples are labeled as negative.
- **Prediction**: Then when you want to classify an instance, you get the decision score from each classifier for that instance, and you select the class whose classifier outputs the highest score.



One-versus-all (OvA) - Example

- One way to create a system that can classify the digit images into 10 classes (from 0 to 9) is to train 10 binary classifiers, one for each digit (a 0-detector, a 1-detector, a 2-detector, and so on).
- Then when you want to classify an image, you get the decision score from each classifier for that image, and you select the class whose classifier outputs the highest score

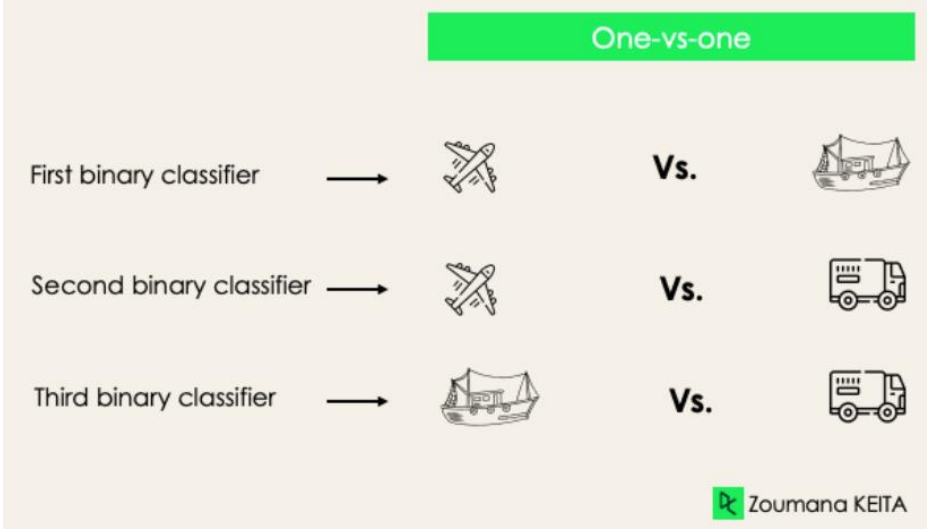


one-versus-one (OvO)

- To train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on.
- If there are **N classes**, you need to train **N × (N – 1) / 2** classifiers.
- $\binom{10}{2} = \frac{10!}{8!\ 2!} = \frac{10 \times 9}{2} = 45$
- For the MNIST problem, this means training **45 binary classifiers**! When you want to classify an image, you have to run the image through all 45 classifiers and see which class wins the most duels

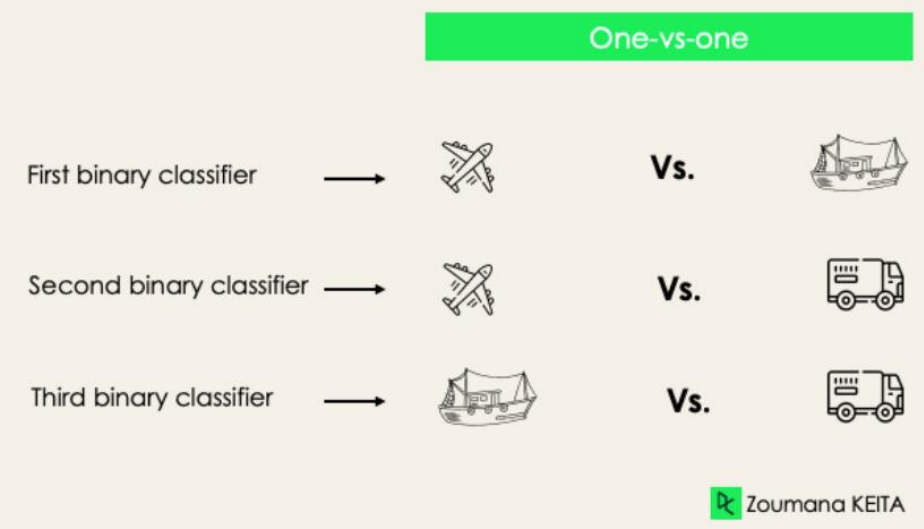
one-versus-one (OvO)

- Each classifier is trained on a single binary dataset, and the final class is predicted by a majority vote between all the classifiers.
- The main advantage of OvO is that each classifier only needs to be trained on the part of the training set containing the two classes that it must distinguish.



one-versus-one (OvO)

- Some algorithms (such as support vector machine classifiers) scale poorly with the size of the training set.
- For these algorithms OvO is preferred because it is faster to train many classifiers on small training sets than to train few classifiers on large training sets.
- For most binary classification algorithms, however, OvR is preferred.



Note:

- Scikit-Learn detects when you try to use a binary classification algorithm for a multiclass classification task, and it automatically runs OvR or OvO, depending on the algorithm
  - SVM - OvO.
  - Others - OVA
- If you want to force ScikitLearn to use one-versus-one or one-versus-all, you can use the OneVsOneClassifier or OneVsRestClassifier classes

Multiclass Example using SVM

- SVMs do not scale well to large datasets, so let's only train on the first 2,000 instances, or else this section will take a very long time to run.

```
from sklearn.svm import SVC

svm_clf = SVC(random_state=42)
svm_clf.fit(X_train[:2000], y_train[:2000])  # y_train, not y_train_5
```

- Since there are 10 classes (i.e., more than 2), Scikit-Learn used the OvO strategy and trained 45 binary classifiers

```
svm_clf.predict([some_digit])
```

```
array(['5'], dtype=object)
```

Multiclass Example using SVM

- This code actually made **45 predictions**—one per pair of classes—and it **selected the class that won the most duels**. If you call the decision_function() method, you will see that it returns 10 scores per instance: one per class

```
some_digit_scores = svm_clf.decision_function([some_digit])
some_digit_scores.round(2)
```

```
array([[ 3.79,  0.73,  6.06,  8.3 , -0.29,  9.3 ,  1.75,  2.77,  7.21,
         4.82]])
```

```
class_id = some_digit_scores.argmax()
class_id
```

```
5
```

```
svm_clf.classes_
```

```
array(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'], dtype=object)
```

```
svm_clf.classes_[class_id]
```

```
'5'
```

Multiclass Example using SVM

- If you want to force ScikitLearn to use one-versus-one or one-versus-all, you can use the **OneVsOneClassifier** or **OneVsRestClassifier** classes

```
from sklearn.multiclass import OneVsRestClassifier

ovr_clf = OneVsRestClassifier(SVC(random_state=42))
ovr_clf.fit(X_train[:2000], y_train[:2000])
```

- Let's make a prediction, and check the number of trained classifiers:

```
ovr_clf.predict([some_digit])
```

```
array(['5'], dtype='<U1')
```

```
len(ovr_clf.estimators_)
```

```
10
```

Multiclass Example using SGDClassifier

- Training an SGDClassifier on a multiclass dataset and using it to make predictions

```
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train)
sgd_clf.predict([some_digit])
```

```
array(['3'], dtype='<U1')
```

- Prediction errors do happen!
- This time Scikit-Learn used the OvR strategy under the hood: since there are 10 classes, it trained 10 binary classifiers

Multiclass Example using SGDClassifier

- The decision_function() method now returns one value per class.

```
sgd_clf.decision_function([some_digit]).round()
```

```
array([[-31893., -34420.,  -9531.,   1824., -22320.,  -1386., -26189.,
        -16148.,  -4604., -12051.]])
```

- You can see that the classifier is not very confident about its prediction: almost all scores are very negative, while class 3 has a score of +1,824, and class 5 is not too far behind at –1,386

Multiclass Example using SGDClassifier

- Of course, you'll want to evaluate this classifier on more than one image. Since there are roughly the same number of images in each class, the accuracy metric is fine

```
cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

```
array([0.87365, 0.85835, 0.8689 ])
```

- It gets over 85.8% on all test folds. If you used a random classifier, you would get 10% accuracy, so this is not such a bad score, but you can still do much better

Multiclass Example using SGDClassifier

- Simply scaling the inputs increases accuracy above 89.1%:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.astype("float64"))
cross_val_score(sgd_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
```

```
array([0.8983, 0.891 , 0.9018])
```

Next Lecture

- Here, we will assume that you have found a promising model and you want to find ways to improve it. One way to do this is to **analyze the types of errors** it makes.
- Error Analysis

Next Lecture

- **Multilabel classification**—A classification task where each input sample can be assigned multiple labels. For instance, a given image may contain both a cat and a dog and should be annotated both with the “cat” label and the “dog” label. The number of labels per image is usually variable.

Next Lecture

- classify **Reuters newswires** into **46 mutually exclusive topics**. Because we have many classes, this problem is an instance of **multiclass classification**, and because each data point should be classified into **only one category**, the problem is more specifically an instance of **single-label multiclass classification**.
- If each data point could belong to **multiple categories** (in this case, topics), we'd be facing a **multilabel multiclass classification problem**.

Next Lecture

- Reuters dataset, a set of short newswires and their topics, published by Reuters in 1986. It's a simple, widely used toy dataset for text classification. There are 46 different topics; some topics are more represented than others, but each topic has at least 10 examples in the training set.
- Like IMDB and MNIST, the Reuters dataset comes packaged as part of Keras