Object Recognition (Image Based) using Neural Network

Group Members

Abhyank Panwar

Madhuri Dilliker

Gipson Selvakumar Nadar

Abstract

- In this project, we used the CIFAR-10 object recognition dataset as a benchmark to implement a recently published deep neural network.

- This project is build using Keras a high level neural application programming interface that supports both Tensorflow and Theona backends.

- We import dataset from Keras, use one hot vectors for categorical labels, add layers to Keras model, load pre trained weights and make predictions using pre trained Keras model.

Dataset

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. * There are 50000 training images and 10000 test images.
- The dataset is divided into five training batches and one test batch, each with 10000 images.

Dataset

- These are the classes in the dataset:

  - Airplane
  - Automobile
  - Bird
  - Cat
  - Horse
  - Truck
  - Deer
  - Dog
  - Frog
  - Ship

  - We have training set of 50000 images and test set of 10000 images

Approach

- Downloaded dataset
- Scaling the dataset
- Applied PCA
- KNN
- Random Forest
- SVM
- Evaluation of different models
- Convolution Neural Network
- Conclusion

Keras

- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.
- Use Keras if you need a deep learning library that:
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Tensorflow

TensorFlow is an open source software library released in 2015 by Google to make it easier for developers to design, build, and train deep learning models.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*.

Scaling the data

- Scaling is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.
- In other words, we put all the pixel data in one line and make connections with the final layer. And once again. What is the final layer for? The classification of 'the cats and dogs.'

Principal Component Analysis

- A more common way of speeding up a machine learning algorithm is by using Principal Component Analysis (PCA).
- If your learning algorithm is too slow because the input dimension is too high, then using PCA to speed it up can be a reasonable choice. This is probably the most common application of PCA. Another common application of PCA is for data visualization.

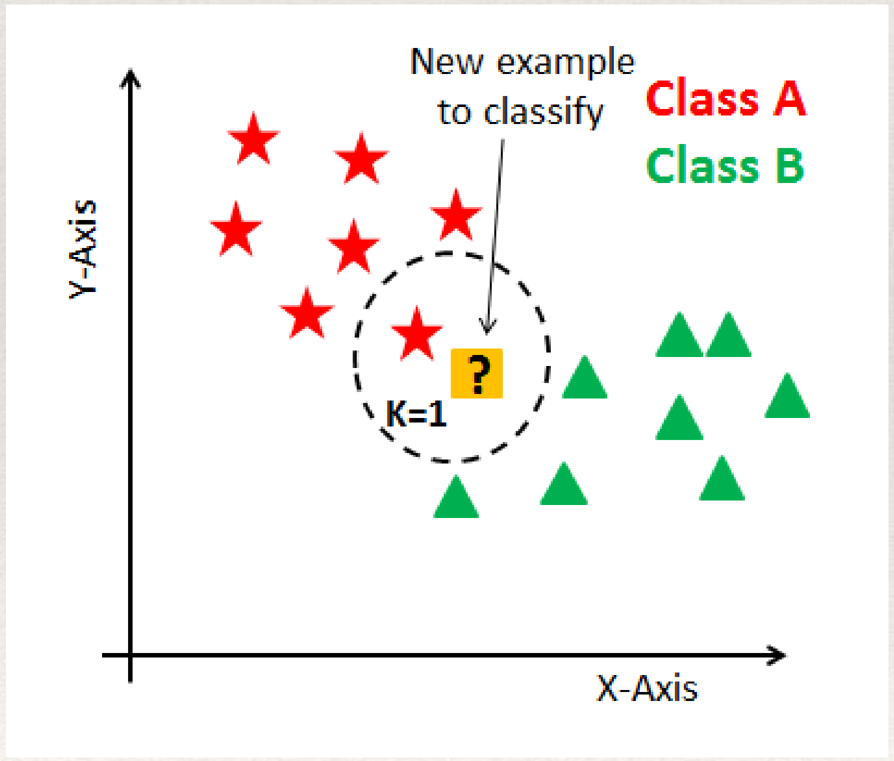
K-Nearest Neighbor

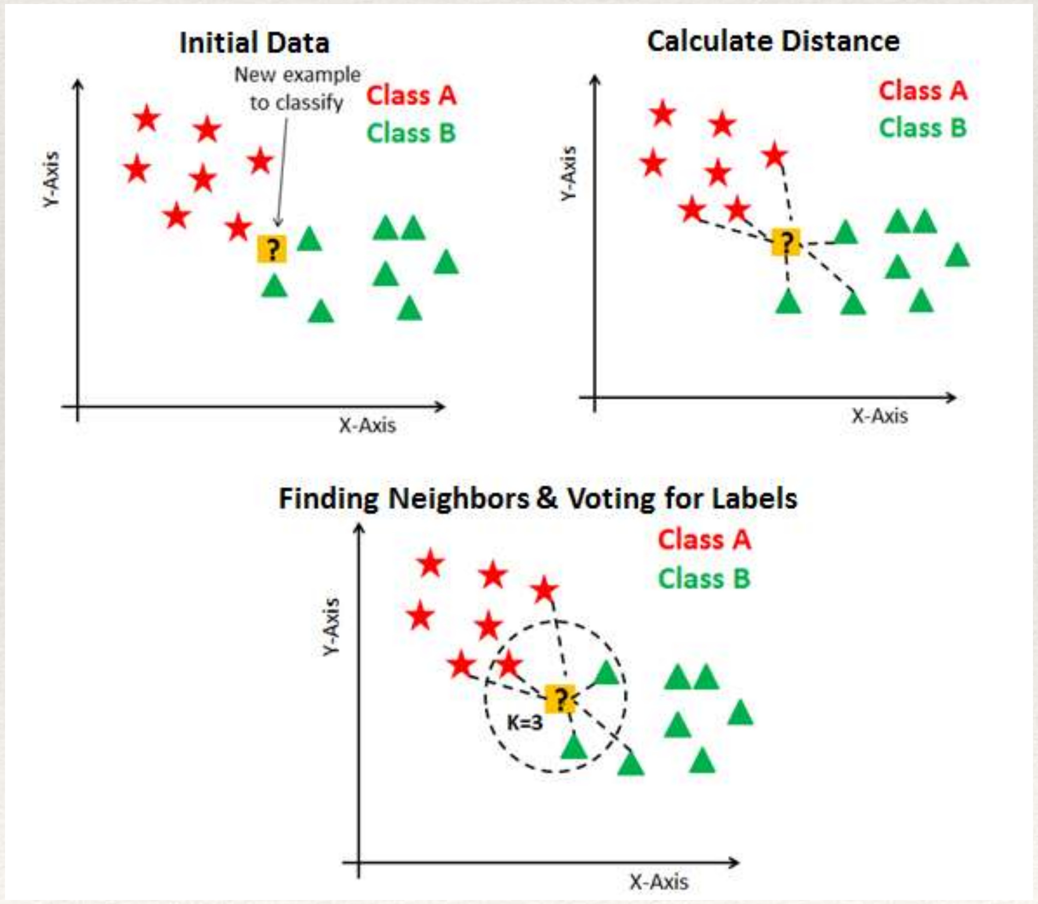
- K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms.
- KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition.
- KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution.

How does the KNN algorithm work?

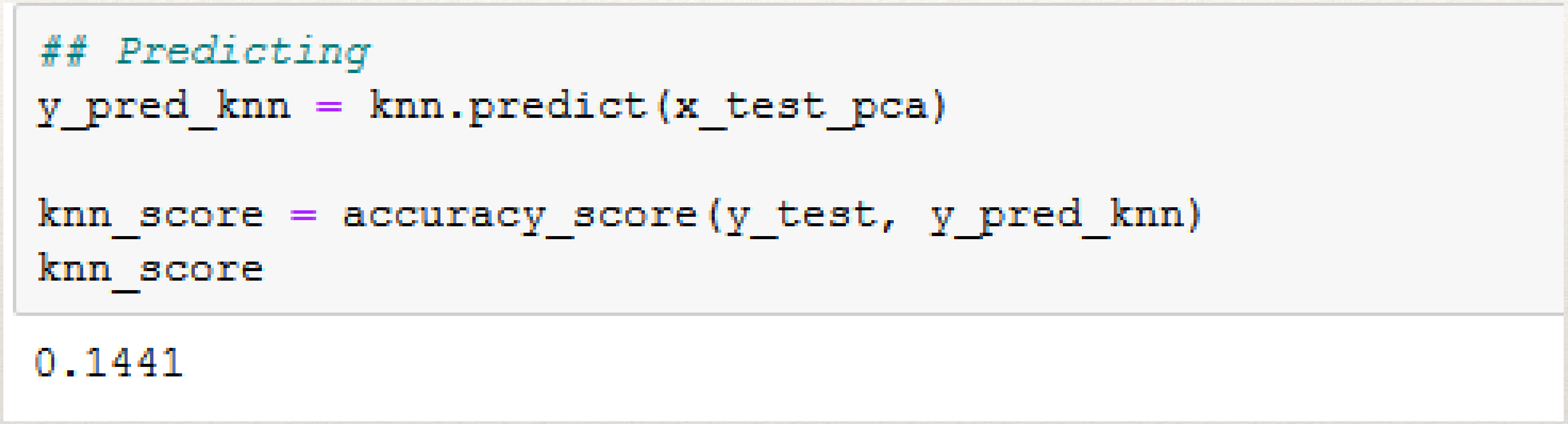
- In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor.
- K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm.

- KNN has the following basic steps:
  - Calculate distance
  - Find closest neighbors
  - Vote for labels





KNN Result



```
## Predicting
y_pred_knn = knn.predict(x_test_pca)

knn_score = accuracy_score(y_test, y_pred_knn)
knn_score
```

```
0.1441
```

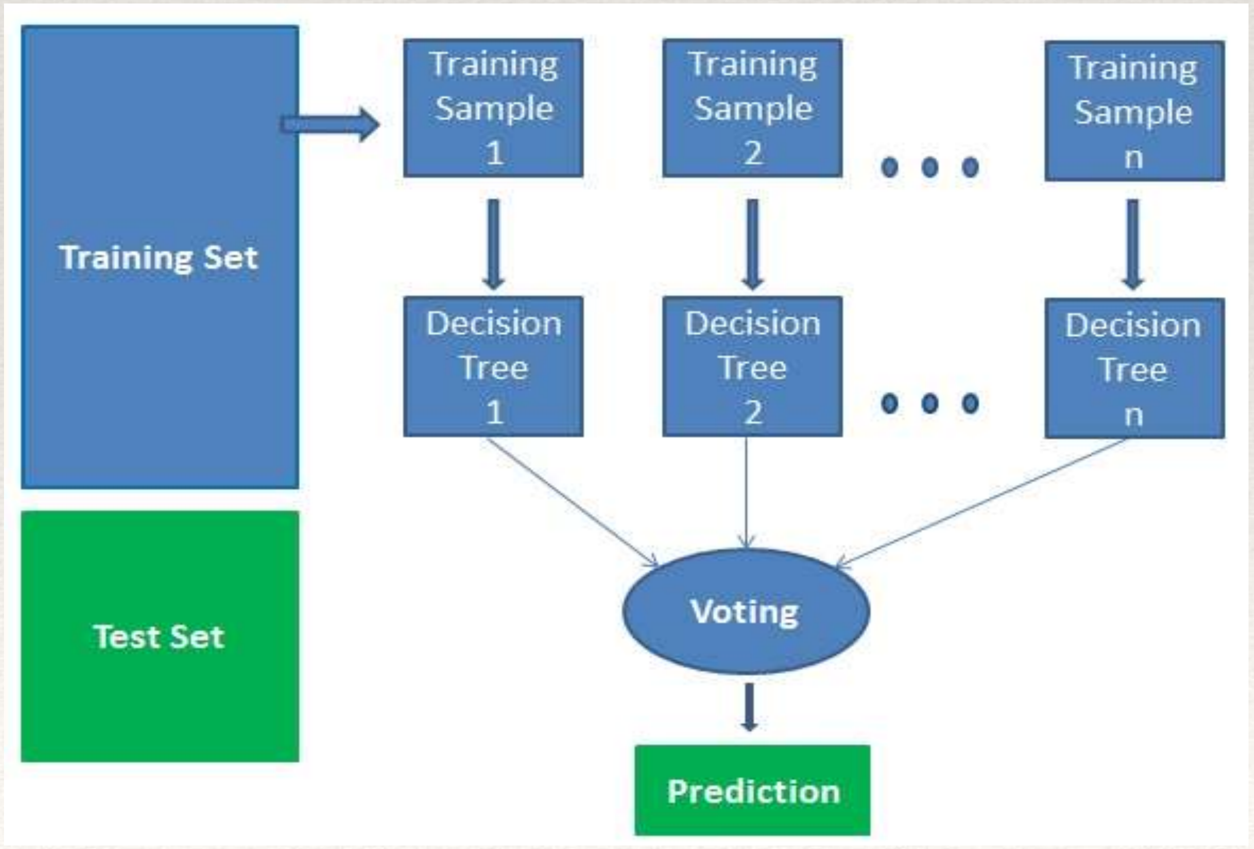
Random Forest

- Random forests is a supervised learning algorithm. It can be used both for classification and regression.
- Random forests has a variety of applications, such as recommendation engines, image classification and feature selection.

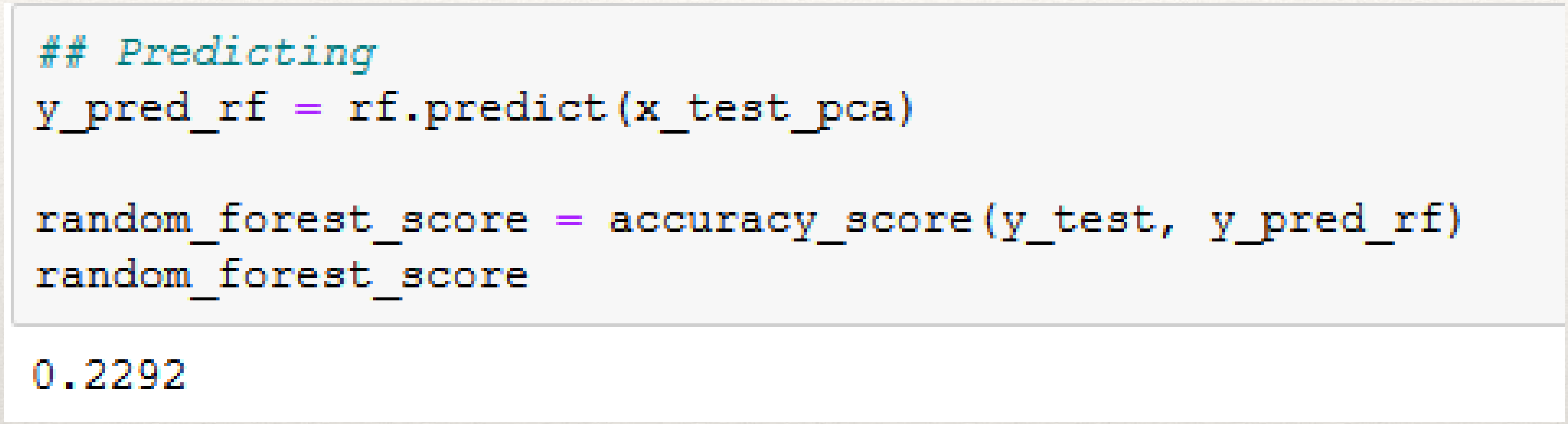
How does the random forest work?

- It works in four steps:
  - Select random samples from a given dataset.
  - Construct a decision tree for each sample and get a prediction result from each decision tree.
  - Perform a vote for each predicted result.
  - Select the prediction result with the most votes as the final prediction.

- Some advantages
  - Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
  - It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
  - The algorithm can be used in both classification and regression problems.
  - Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
  - You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.



Random Forest Result



```
## Predicting
y_pred_rf = rf.predict(x_test_pca)

random_forest_score = accuracy_score(y_test, y_pred_rf)
random_forest_score
```

```
0.2292
```

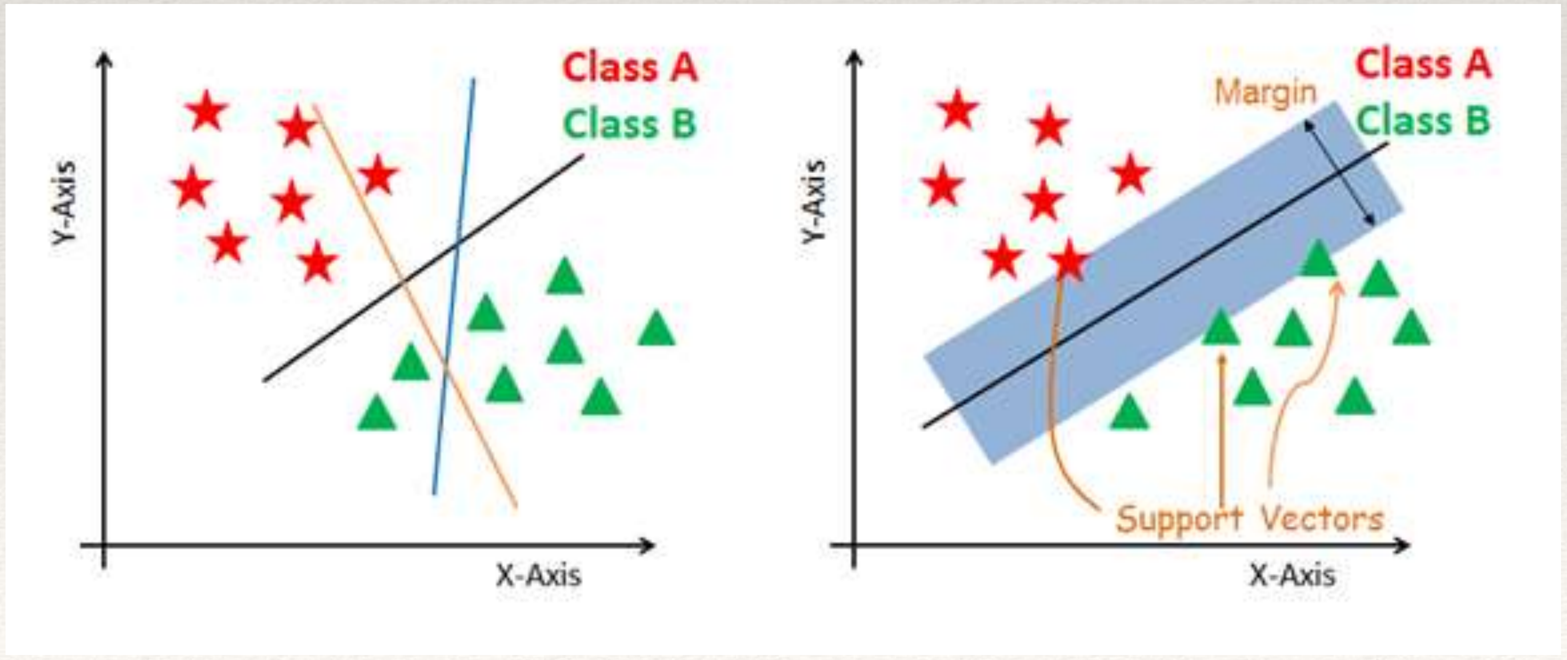
Support Vector Machine(SVM)

- SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces.
- It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

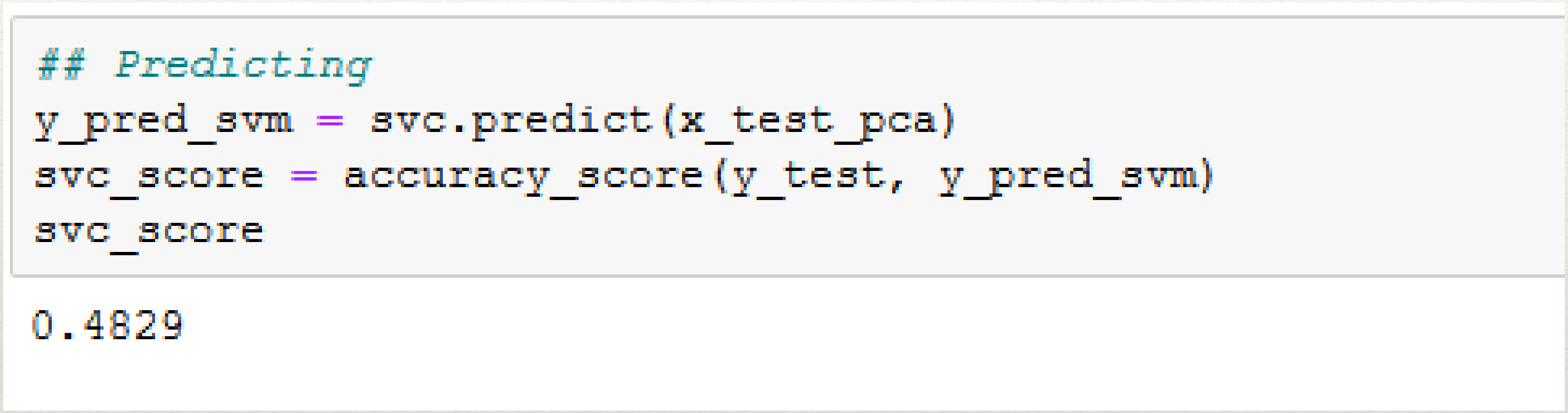
How does it work?

- Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables.
- SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

- The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin.
- The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:
    - Generate hyperplanes which segregates the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.
    - Select the right hyperplane with the maximum segregation from the either nearest data points as shown in the figure.



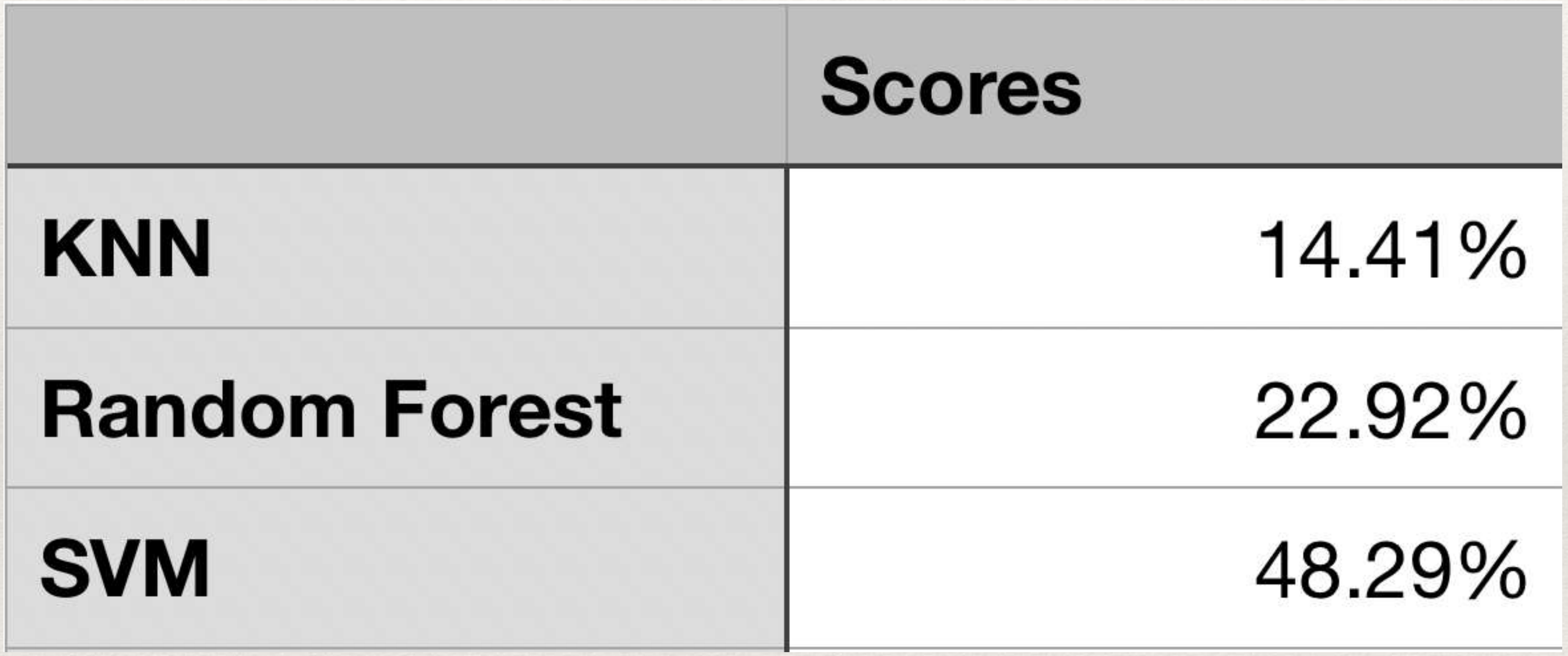
SVM Result



```
## Predicting
y_pred_svm = svc.predict(x_test_pca)
svc_score = accuracy_score(y_test, y_pred_svm)
svc_score
```

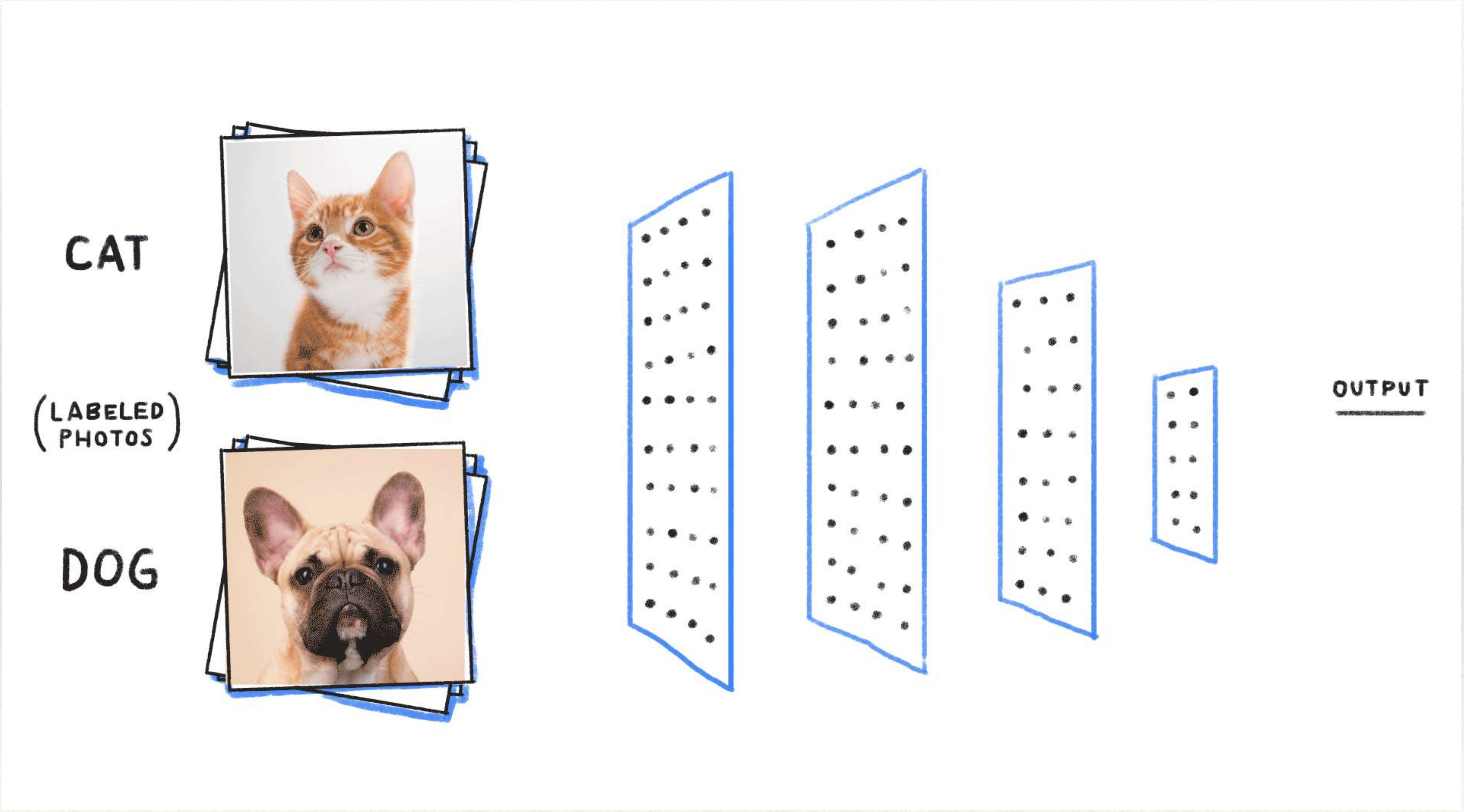
```
0.4829
```

Evaluation of different models

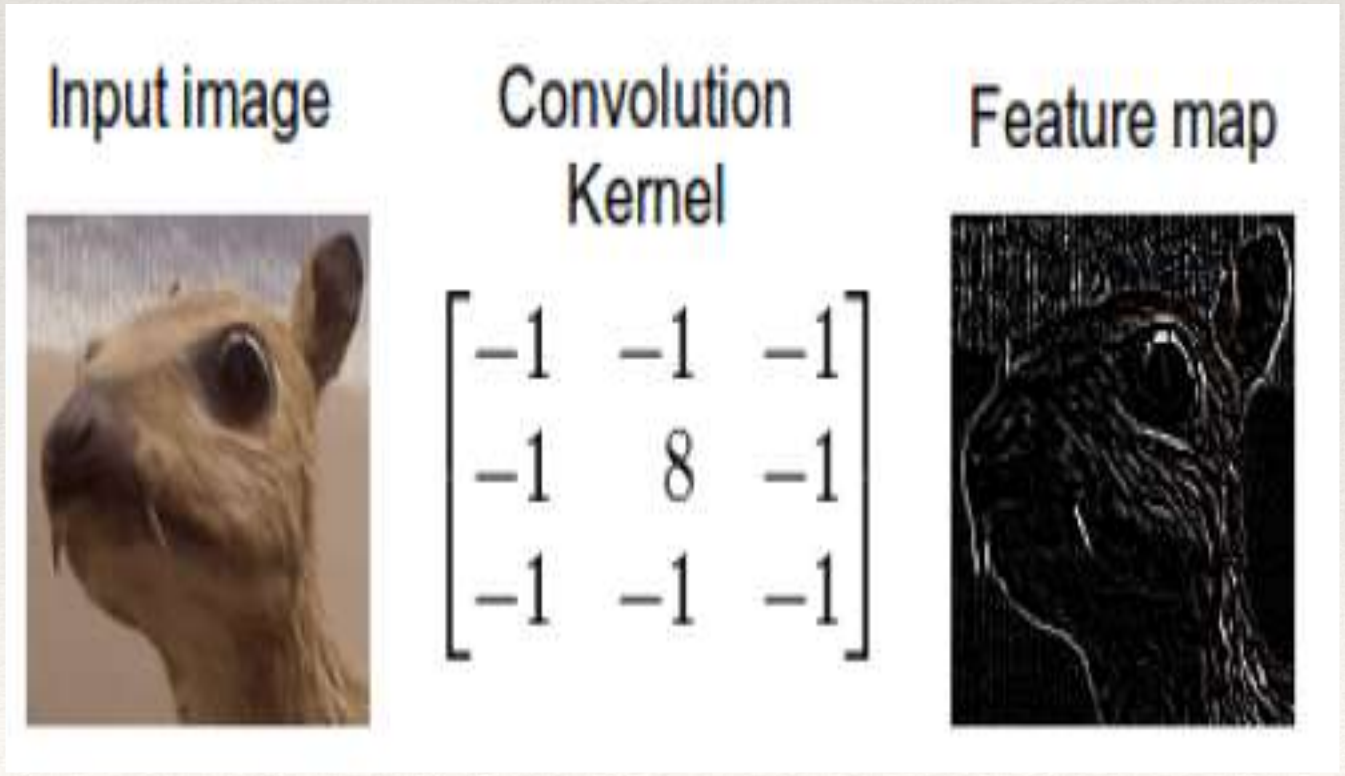


	Scores
KNN	14.41%
Random Forest	22.92%
SVM	48.29%

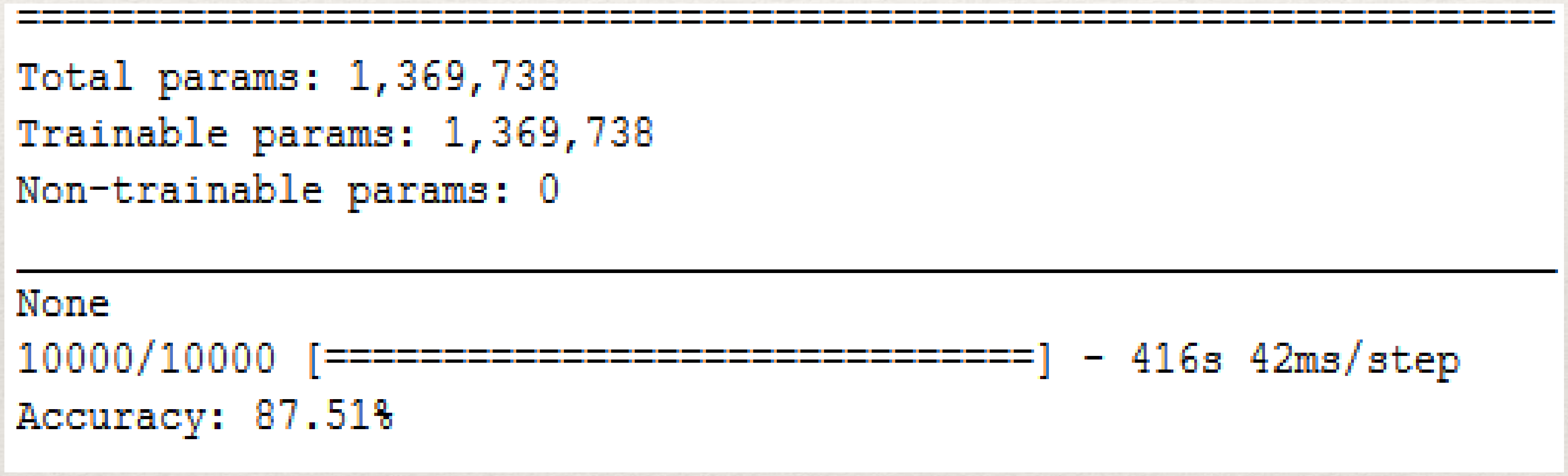
CNN (Convolutional Neural Network)!



- By definition, Convolution is a mathematical operation on two objects to produce an outcome that expresses how the shape of one is modified by the other. With this computation, we detect a particular feature from the input image and get the result having information about that feature. This is called ‘feature map.’ If we see this with the real image example, the outcome is shown below.


Input image
Convolution Kernel
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
Feature map

CNN Model Accuracy



```
==================================================================
Total params: 1,369,738
Trainable params: 1,369,738
Non-trainable params: 0

__________________________________________________________________
None
10000/10000 [==============================] - 416s 42ms/step
Accuracy: 87.51%
```

Conclusion

- For machine learning models the accuracy for SVM was good enough as compared to other models.
- But after training and testing the model using CNN, accuracy for correct classification of image samples is quite high i.e (87.51%) after using 350 epoch
- Only Drawback is that it takes lot of time to run the epoch

Thank You

Any Questions?