In [1]:

```
import cifar10
import numpy as np
```

In [2]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.decomposition import PCA

from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

Importing Datsets

In [6]:

```
cifar10.data_path = "data/CIFAR-10/"
```

In [7]:

```
cifar10.maybe_download_and_extract()
```

```
Data has apparently already been downloaded and unpacked.
```

In [8]:

```
class_names = cifar10.load_class_names()
class_names
```

```
Loading data: data/CIFAR-10/cifar-10-batches-py/batches.meta
```

Out[8]:

```
['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']
```

In [9]:

```
images_train, cls_train, labels_train = cifar10.load_training_data()
images_test, cls_test, labels_test = cifar10.load_test_data()
```

```
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_1
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_2
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_3
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_4
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_5
Loading data: data/CIFAR-10/cifar-10-batches-py/test_batch
```

Analysing the data

In [10]:

```
## It represents the RGB values of the pixel(12,22) of 5th image in training dataset
images_train[5][12][22]
```

Out[10]:

```
array([0.78823529, 0.28235294, 0.27058824])
```

Flattening the image data

In [11]:

```
images_train.shape
```

Out[11]:

```
(50000, 32, 32, 3)
```

In [12]:

```
images_test.shape
```

Out[12]:

```
(10000, 32, 32, 3)
```

In [13]:

```
x_train = images_train.reshape(images_train.shape[0],-1)
x_test = images_test.reshape(images_test.shape[0], -1)
```

In [14]:

```
x_train.shape
```

Out[14]:

```
(50000, 3072)
```

In [15]:

```
x_test.shape
```

Out[15]:

```
(10000, 3072)
```

In [16]:

```
y_train = cls_train
y_test = cls_test
```

In [17]:

```
labels_train
```

Out[17]:

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.]])
```

Applying PCA

In [18]:

```
x_train.shape
```

Out[18]:

```
(50000, 3072)
```

In [19]:

```
pca = PCA()
pca.fit_transform(x_train)
```

Out[19]:

```
array([[-6.40101763e+00,  2.72903876e+00,  1.50171052e+00, ...,
        -7.62287826e-04, -1.16359323e-03, -3.49768400e-04],
       [ 8.29783385e-01, -9.49943219e-01,  6.00375260e+00, ...,
         1.08434589e-04,  5.76715387e-04, -7.71005927e-04],
       [ 7.73019978e+00, -1.15221023e+01, -2.75362051e+00, ...,
        -1.04460109e-03, -2.14105740e-04, -3.93469526e-04],
       ...,
       [ 7.07346587e-01, -1.13425212e+01, -3.93914878e-01, ...,
        -3.80341568e-04,  1.86432348e-03,  1.44785229e-03],
       [ 1.13679090e+01, -3.37597765e+00,  5.56619766e+00, ...,
        -4.67096832e-04,  7.44972963e-04, -1.67065056e-03],
       [ 4.19279645e+00, -1.18418848e+00, -4.44596297e+00, ...,
         9.67968258e-04,  2.40363971e-04, -2.80506224e-05]])
```

In [20]:

```
pca.explained_variance_.shape
```

Out[20]:

(3072,)

In [21]:

```
# Calculating optimal k to have 95% (say) variance

k = 0
total = sum(pca.explained_variance_)
current_sum = 0

while(current_sum / total < 0.99):
    current_sum += pca.explained_variance_[k]
    k += 1
k
```

Out[21]:

658

In [22]:

```
## Applying PCA with k calcuated above

pca = PCA(n_components=k, whiten=True)

x_train_pca = pca.fit_transform(x_train)
x_test_pca = pca.transform(x_test)
```

Prediction using Random Forest

In [42]:

```
pca
```

Out[42]:

```
PCA(copy=True, iterated_power='auto', n_components=658, random_state=None,
  svd_solver='auto', tol=0.0, whiten=True)
```

In [43]:

```
## Training
rf = RandomForestClassifier()
rf.fit(x_train_pca, y_train)
```

```
C:\Users\jscmp4\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Future
Warning: The default value of n_estimators will change from 10 in version 0.20 to
100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[43]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [44]:

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

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

Out[44]:

```
0.2349
```

Prediction using KNN

In [45]:

```
## Training
knn = KNeighborsClassifier()
knn.fit(x_train_pca, y_train)
```

Out[45]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [46]:

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

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

Out[46]:

```
0.1432
```

Prediction using Logistic Regression

In [23]:

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegressionCV
```

In [27]:

```
clf = LogisticRegressionCV(cv=7, random_state=0,
                           multi_class='multinomial').fit(x_train_pca, y_train)
clf.predict(x_train_pca)

clf.predict_proba(x_train_pca).shape

clf.score(x_train_pca, y_train)
```

Out[27]:

```
0.45232
```

In [47]:

```
## Training
lr = LogisticRegression()
lr.fit(x_train_pca, y_train)
```

```
C:\Users\jscmp4\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
C:\Users\jscmp4\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the
multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[47]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [48]:

```
## Predicting
y_pred_lr = lr.predict(x_test_pca)
logistic_regression_score = accuracy_score(y_test, y_pred_lr)
logistic_regression_score
```

Out[48]:

```
0.4007
```

Prediction using SVM

In [35]:

```
## Training
svc = svm.SVC()
svc.fit(x_train_pca, y_train)
```

Out[35]:

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

In [36]:

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

Out[36]:

0.4834

Till Now, Best accuracy comes from Support vector classifier