Datasets:

1. Image dataset: GTSRB - German Traffic Sign Recognition Benchmark (From Kaggle)
2. Non-image dataset: Mobile Price Classification (From previous course)
3. Self-made dataset: 飲料辨識

Algorithms:

1. KNN
2. Decision Tree
3. Random Forest
4. SVM (SVC and NuSVC)
5. Lenet

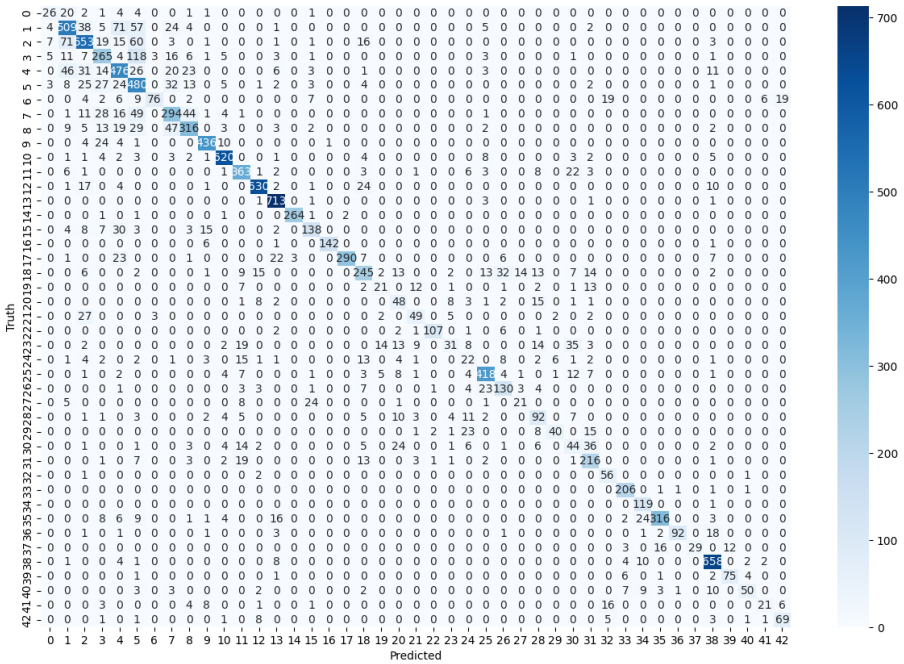
Analysis:

1. GTSRB:

GTSRB 是 German Traffic Sign Recognition Benchmark 的縮寫，要辨識出德國的各種交通號誌，是 IJCNN 2011 所使用的題目，而這個資料庫是一個單照片多分類(single-image, multi-class)的分類問題，裡面有超過 50,000 張的照片和 43 種類別，而且分類的內容是一個現實世界可能會遇到的問題。

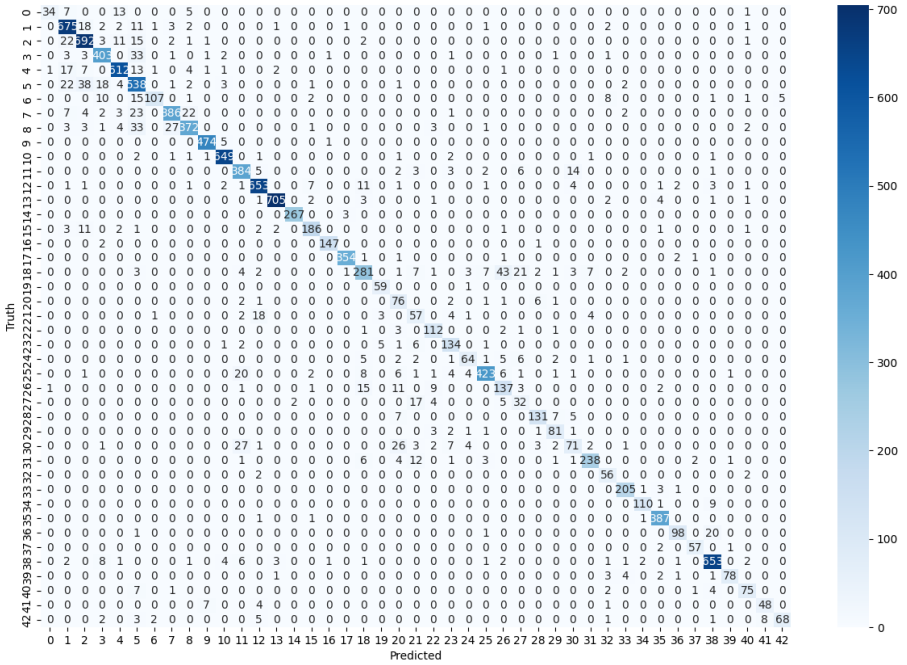
a. NuSVC:

NuSVC 是 SVM 再加上一個限制，限制訓練誤差和分數的下界，這邊我的 nu 設定為 0.1，kernel 為 rbf，gamma=0.001。訓練一次需要，訓練出來的準確率有 0.773，但訓練一次就需要 18 分鐘，非常沒有效率。



b. Lenet:

Lenet 是 Yann LeCun 團隊於 1998 年提出的一個 CNN 架構，有兩個卷積層，兩個 pooling 層和池化層、全連接卷積層、全連接層所組成，訓練時間比 NuSVC 還要快上許多，訓練 20 個 epochs 也只需要約 1 分鐘，得到的準確率維 0.862，比 NuSVC 來的好很多，也難怪後來在處理影像上，CNN 會越來越成為主流。



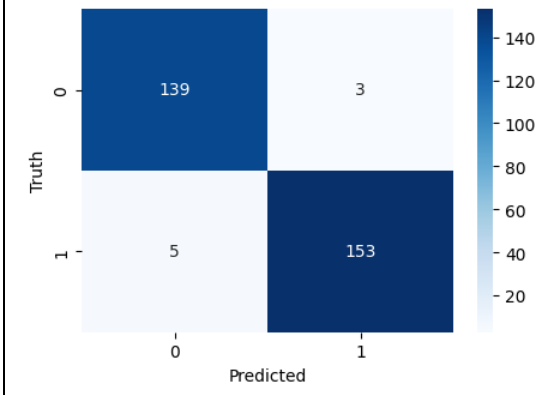
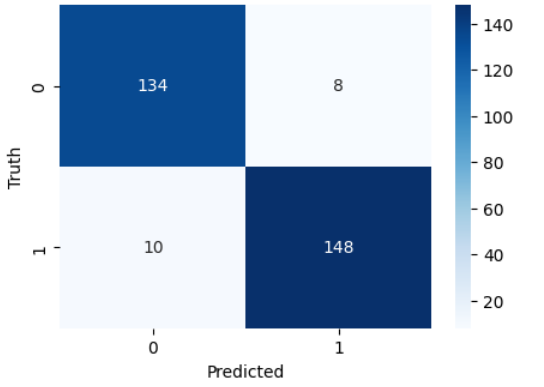
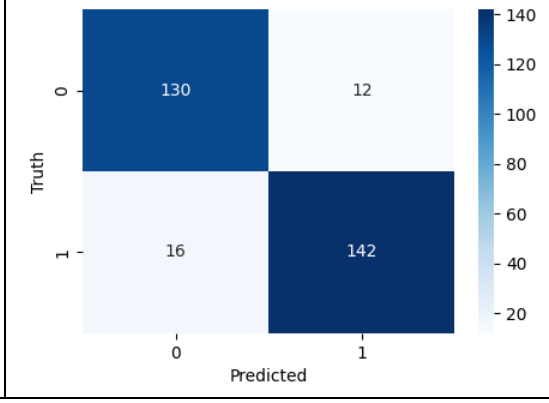
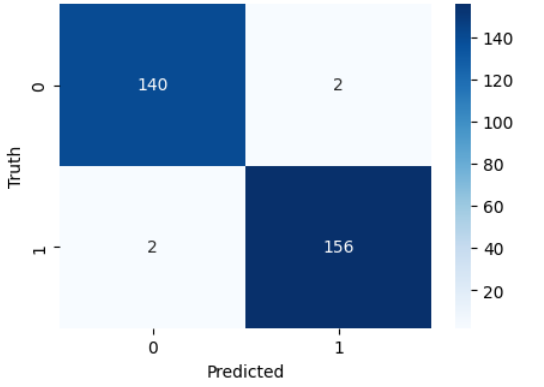
c. Discussion:

根據這兩個分類器來說，可以預期 CNN based 的方法一定比 SVM based 的方法 performance 來的更好，但我沒有想到會差距這麼大，我猜是因為 SVM 要一次看 3072 維的資料(32*32*3)，要想辦法去找 hyperplane 來切割出這 40 多種的分類需要花很多的時間，如果有時間應該可以試試加上 PCA 去降維度，加入的同時就可以額外去做 Grid Search，讓 performance 可以更好。

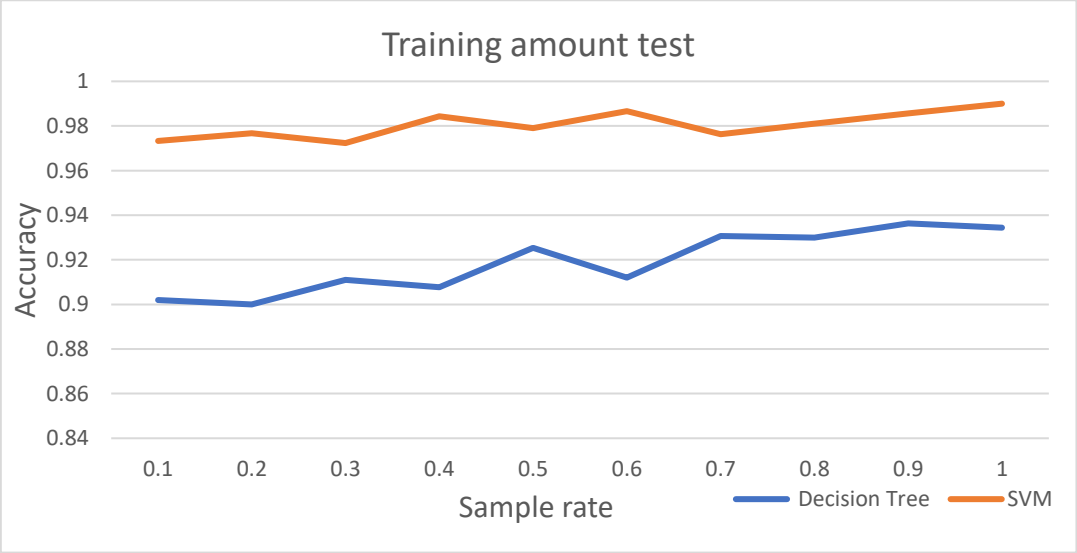
2. Mobile Price Classification:

Mobile Price Classification 在 Kaggle 上也有類似的資料庫，但 Kaggle 的沒有提供 Test set 的正確解答，我的 dataset 是從先前的課程所提供的 dataset，在 Mobile Price Classification 的地方會做 Grid Search，找出最佳 parameter 時順便做 cross validation (cv=5)，也有做 training data 數量的測試，分類器選 Decision Tree 和 SVM，從 sample rate 0.1, 0.2, ..., 1，每個測試 3 次取平均。

a. Compare the results when using different classifiers.

Classifier	KNN	Decision Tree
Accuracy	0.973	0.94
Confusion Matrix		
Classifier	Random Forest	SVM
Accuracy	0.907	0.987
Confusion Matrix		

b. Compare the results when using different amounts of training data.



c. Discussion

可以看到 a 部分的準確率排名是 SVM > KNN > Decision Tree > Random Forest，而訓練時間基本上都是瞬間完成，所以不好比較，而這個 dataset 應該是

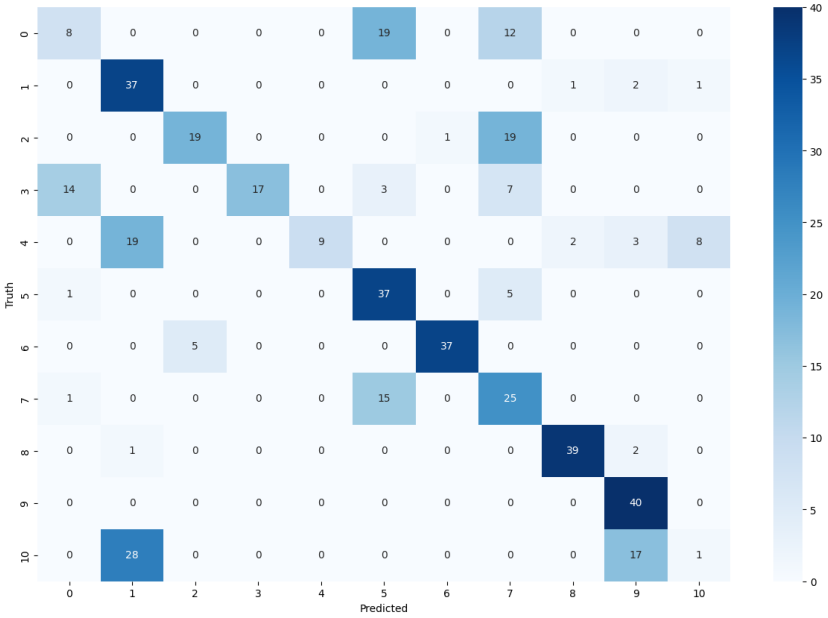
比較單純，所以光是用 KNN 就可以拿到非常好的成績了，SVM 也能很好的去分類，至於 Decision Tree 和 Random Forest 的 performance 不好，我猜是 over fitting 的關係，導致太貼近訓練資料，這應該也能解釋 Random Forest 比 Decision Tree 還要更不準的原因。再來看不同訓練數量的測試，可以看到 SVM 一直都在 Random Forest 的上方，而且浮動的幅度也比 Random Forest 還要小，可以說 SVM 對訓練資料量的要求小很多。因為時間的關係我只做了 Decision Tree 和 Random Forest，有時間的話還可以補上 SVM 和 KNN，或許會有一些有趣的發現和結果。

3. 飲料辨識:

這個 Dataset 是我和高鈺鴻(學號 109550040)一起製作的，透過拍各種飲料瓶的四周來當作資料庫，我們總共拍了 11 種罐狀物品，每個物品大約有 50 張的照片，所以總共有大約 550 張照片做成我們的資料庫，裡面的類別有: Bar、日式綠茶、淡麗、麥香、綠茶、藍莓酒、寶礦力、蘋果酒、鐵觀音、水壺和刮鬍泡，有些是同品牌的飲料所以我們認為某些辨識上可能會比較複雜。而我使用了和第一個 Image Dataset 同樣的兩種 classifier，NuSVC 和 Lenet。

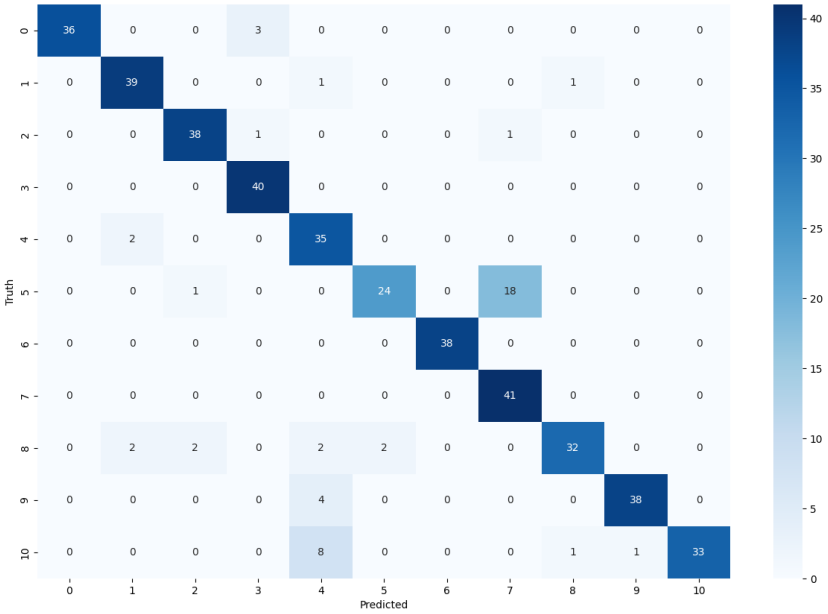
a. NuSVC:

下圖是 NuSVC 的 Confusion Matrix，我把資料庫的 20%當成 validation set，3 次得到的準確率是 0.591、0.594、0.673，平均下來是 0.619。



b. Lenet:

下圖是 Lenet 的 Confusion Matrix，同樣把資料庫的 20%當成 validation set，3 次得到的準確率是 0.887、0.910、0.872，平均下來是 0.890。



c. Discussion

可以從兩張圖和得到的準確率來知道 CNN 在一次的贏過了 NuSVC 許多，NuSVC 只要顏色像或是輪廓像就會被誤判，但 CNN 只有在同品牌且只有些微文字和顏色不同的鋁罐才有出現比較大量的錯誤。如果還有更多時間的話，我應該會再繼續加大資料庫的種類和張數，甚至可以拍更多奇怪角度的照片，讓整個辨識的能力能夠更好。下面有資料庫每個類別的範例:

Code:

1. GTSRB – NuSVC:

```
import numpy as np
import pandas as pd
from PIL import Image
from matplotlib import pyplot as plt
import seaborn as sns
from skimage.io import imread
import cv2
from sklearn import preprocessing
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import GridSearchCV
# load train and test data
train = pd.read_csv('Train.csv')
train_x = []
for i in train['Path']:
    img = Image.fromarray(cv2.imread(i), 'RGB')
    train_x.append(np.array(img.resize((32,32))))
train_x = np.array(train_x)
train_y = np.array(train['ClassId'].values)
print(train_x.shape)
test = pd.read_csv('Test.csv')
test_x = []
for i in test['Path']:
    img = Image.fromarray(cv2.imread(i), 'RGB')
    test_x.append(np.array(img.resize((32,32))))
test_x = np.array(test_x)
test_y = np.array(test['ClassId'].values)
print(test_x.shape)
# normalize data
train_x.resize(train_x.shape[0], 32 * 32 * 3)
test_x.resize(test_x.shape[0], 32 * 32 * 3)
train_x = preprocessing.normalize(train_x)
test_x = preprocessing.normalize(test_x)
# train model
```

```
svm_clf = svm.NuSVC(nu=0.1, kernel='rbf', gamma=0.001, random_state=8, verbose=10)
svm_clf.fit(train_x, train_y)
svm_clf.score(test_x, test_y)
# predict
svm_pred = svm_clf.predict(test_x)
# plot confusion matrix
plt.figure(figsize=(15, 10))
sns.heatmap(confusion_matrix(test_y, svm_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

2. GTSRB – CNN:

```
from sklearn.metrics import accuracy_score, confusion_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
# prepare the data
data = []
labels = []
classes = 43
cur_path = os.getcwd()
print(cur_path)
# Load the images and lables from the dataset
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
```

```
            image = image.resize((32,32))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
# Convert the data and labels into numpy arrays
data = np.array(data)
labels = np.array(labels)
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.25,
random_state=8)
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
# Building the model
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5, 5), input_shape=(32, 32, 3),
activation='tanh'))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5, 5), activation='tanh'))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='tanh'))
model.add(Dense(84, activation='tanh'))
model.add(Dense(43, activation='softmax'))
# Compile and train the model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
epochs = 20
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))
# Plot the loss and accuracy curves for training and validation
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
```

```
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
# Load the test dataset
test = pd.read_csv('Test.csv')
labels = test["ClassId"].values
imgs = test["Path"].values
data=[]
for img in imgs:
    image = Image.open(img)
    image = image.resize((32,32))
    data.append(np.array(image))
X_test=np.array(data)
pred = model.predict(X_test)
pred = np.argmax(pred, axis=1)
print(accuracy_score(labels, pred))
# Plot the confusion matrix
plt.figure(figsize=(15, 10))
sns.heatmap(confusion_matrix(labels, pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

3. MPC – All classifier: (步驟都相同，指註解了 KNN 的部分)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# load the data
```

```
data = pd.read_csv("train.csv")
# split the data into X and y
X = data.drop('price_range', axis=1)
y = data['price_range']
test = pd.read_csv("test.csv")
test_data = test.drop('price_range', axis=1)
test_ans = pd.DataFrame(test['price_range'])
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
# use GridSearchCV to find the best k of KNN
param = {'n_neighbors': np.arange(1, 25)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param, cv=5)
knn_cv.fit(X, y)
# use the best k to train the model
knn_clf = KNeighborsClassifier(n_neighbors=knn_cv.best_estimator_.n_neighbors)
knn_clf.fit(X, y)
# predict the test data
knn_pred = knn_clf.predict(test_data)
print("Accuracy:",metrics.accuracy_score(test_ans, knn_pred))
# plot the confusion matrix
confusion_matrix(test_ans, knn_pred)
plt.figure(figsize=(5, 3.5))
sns.heatmap(confusion_matrix(test_ans, knn_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
param = {'criterion':['gini', 'entropy'], 'max_depth':range(1, 5, 1),
'min_samples_leaf':range(1, 26, 5)}
clf = GridSearchCV(DecisionTreeClassifier(), param, cv=5, scoring='accuracy',
n_jobs=-1, verbose=10)
```

```
clf.fit(X, y)
dt_clf = DecisionTreeClassifier(criterion=clf.best_params_['criterion'],
max_depth=clf.best_params_['max_depth'],
min_samples_leaf=clf.best_params_['min_samples_leaf'])
dt_clf.fit(X, y)
dt_pred = dt_clf.predict(test_data)
print("Accuracy:", accuracy_score(test_ans, dt_pred))
confusion_matrix(test_ans, dt_pred)
plt.figure(figsize=(5, 3.5))
sns.heatmap(confusion_matrix(test_ans, dt_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Random Forest:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
param = {'n_estimators':range(1, 100, 10), 'criterion':['gini', 'entropy'],
'max_depth':range(1, 5, 1), 'min_samples_leaf':range(1, 26, 5)}
clf = GridSearchCV(RandomForestClassifier(), param, cv=5, scoring='accuracy',
n_jobs=-1, verbose=10)
clf.fit(X, y)
rf_clf = RandomForestClassifier(n_estimators=clf.best_params_['n_estimators'],
criterion=clf.best_params_['criterion'], max_depth=clf.best_params_['max_depth'],
min_samples_leaf=clf.best_params_['min_samples_leaf'])
rf_clf.fit(X, y)
rf_pred = rf_clf.predict(test_data)
print("Accuracy:", accuracy_score(test_ans, rf_pred))
confusion_matrix(test_ans, rf_pred)
plt.figure(figsize=(5, 3.5))
sns.heatmap(confusion_matrix(test_ans, rf_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

SVM

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
```

```
param = {'C':[0.001, 0.01, 0.1, 1, 10], 'gamma':[100, 10, 1, 0.1, 0.01],
'kernel':['rbf', 'linear']}
clf = GridSearchCV(SVC(), param, cv=5, scoring='accuracy', n_jobs=-1, verbose=10)
clf.fit(X, y)
SVM_clf = SVC(C=clf.best_params_['C'], gamma=clf.best_params_['gamma'],
kernel=clf.best_params_['kernel'])
SVM_clf.fit(X, y)
svm_pred = SVM_clf.predict(test_data)
print("Accuracy:", accuracy_score(test_ans, svm_pred))
confusion_matrix(test_ans, svm_pred)
plt.figure(figsize=(5, 3.5))
sns.heatmap(confusion_matrix(test_ans, svm_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

4. Self-made dataset – NuSVC:

```
import numpy as np
import pandas as pd
import os
from PIL import Image
from matplotlib import pyplot as plt
import seaborn as sns
from skimage.io import imread
import cv2
from sklearn import preprocessing
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import GridSearchCV
# Loading the data
train_x = []
train_y = []
test_x = []
test_y = []
num_of_classes = 11
for i in range(0, num_of_classes):
    path = './train/' + chr(65 + i) + '/'
    files = os.listdir(path)
    print(len(files))
```

```
    for file in files:
        img = Image.fromarray(cv2.imread(path + file), 'RGB')
        # Split data into train set and val set
        tmp = np.random.randint(0, 100)
        if tmp < 20:
            train_x.append(np.array(img.resize((32,32))))
            train_y.append(i)
        else:
            test_x.append(np.array(img.resize((32,32))))
            test_y.append(i)
train_x = np.array(train_x)
train_y = np.array(train_y)
test_x = np.array(test_x)
test_y = np.array(test_y)
# Normalizing the data
train_x = train_x.reshape(train_x.shape[0], 32*32*3)
test_x = test_x.reshape(test_x.shape[0], 32*32*3)
train_x = preprocessing.normalize(train_x)
test_x = preprocessing.normalize(test_x)
# Training the model
svm_clf = svm.NuSVC(nu=0.1, kernel='rbf', gamma=0.001, random_state=8, verbose=10)
svm_clf.fit(train_x, train_y)
svm_clf.score(test_x, test_y)
# Predicting the model
svm_pred = svm_clf.predict(test_x)
plt.figure(figsize=(15, 10))
sns.heatmap(confusion_matrix(test_y, svm_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

5. Self-made dataset – CNN:

```
from sklearn.metrics import accuracy_score, confusion_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import tensorflow as tf
```

```
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
# Load the data
data = []
labels = []
test = []
test_labels = []
num_of_classes = 11
for i in range(0, num_of_classes):
    path = './train/' + chr(65 + i) + '/'
    files = os.listdir(path)
    print(len(files))
    for file in files:
        image = Image.open(path + file)
        image = image.resize((32,32))
        image = np.array(image)
        # split the data into train and test
        tmp = np.random.randint(0, 100)
        if tmp < 20:
            data.append(image)
            labels.append(i)
        else:
            test.append(image)
            test_labels.append(i)
# Convert the data into numpy array
data = np.array(data)
labels = np.array(labels)
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.25,
random_state=8)
y_train = to_categorical(y_train, num_of_classes)
y_test = to_categorical(y_test, num_of_classes)
# Building the model
model = Sequential()
```

```
model.add(Conv2D(filters=6, kernel_size=(5, 5), input_shape=(32, 32, 3),
activation='tanh'))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5, 5), activation='tanh'))
model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='tanh'))
model.add(Dense(84, activation='tanh'))
model.add(Dense(num_of_classes, activation='softmax'))
# Compile the model and train it
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
epochs = 20
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))
# Plot the accuracy and loss
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()

# Testing the model
val = np.array(test)
val_labels = np.array(test_labels)
cnn_pred = model.predict(val)
```

```
cnn_pred = np.argmax(cnn_pred, axis=1)
print("CNN accuracy: ", accuracy_score(val_labels, cnn_pred))
# plot the confusion matrix
plt.figure(figsize=(15, 10))
sns.heatmap(confusion_matrix(val_labels, cnn_pred), annot=True, cmap='Blues',
fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```