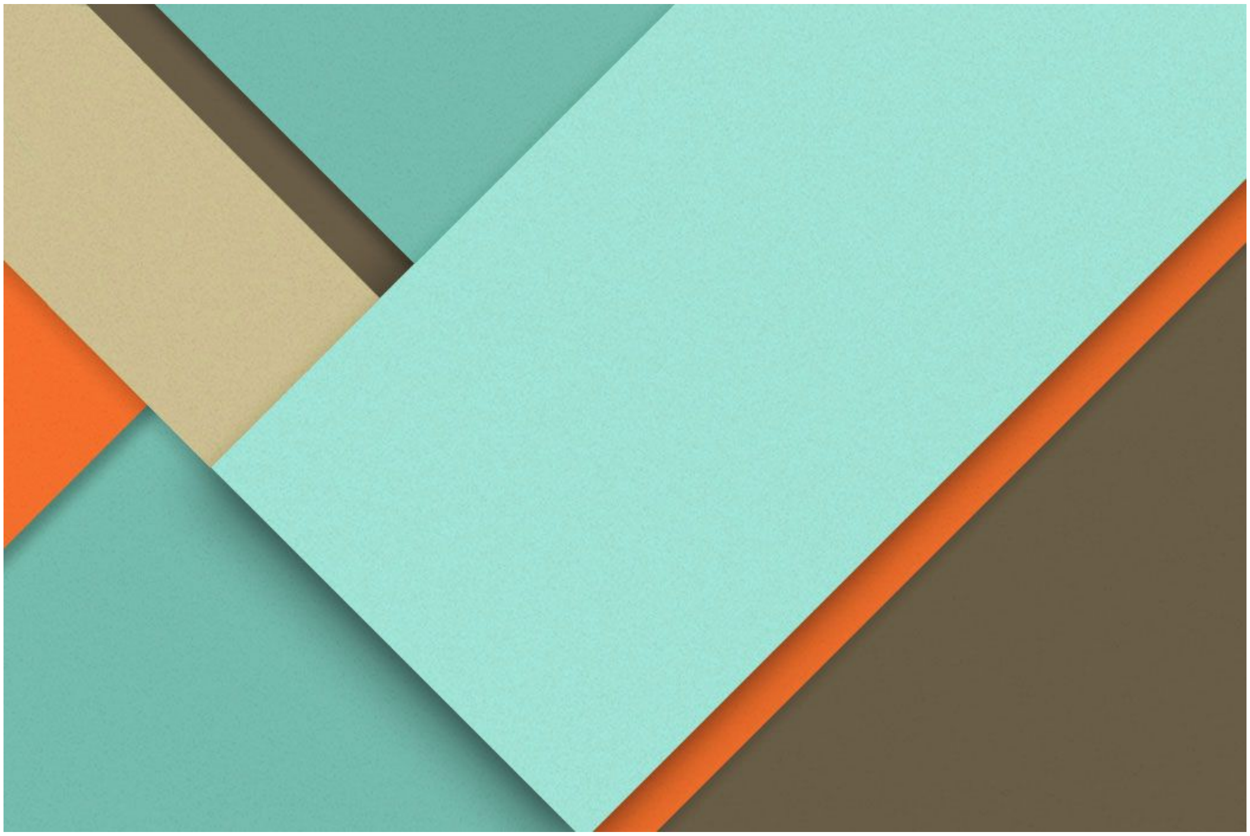


Exercise 3: Digit recognizer task

09.01.2018

Karydis Athanasios
MSc in Data Science
1st Semester
National Centre for Scientific Research 'Demokritos'
University of Peloponnese

Introduction - Problem Definition

In this exercise the goal is to use MNIST dataset, which contains handwritten image and correctly identify digits by evaluating different algorithms. As for the dataset, it contains gray-scale images of hand-drawn digits, from zero through nine. Actually, we are going to use a part of the MNIST dataset which contains 42,000 samples while the original has 70,000 of them.

We are going to use different preprocessing methods and a number of algorithms in order to pick the best model for predictions. Also, we are going to face the problem above as a classification one. To be more specific, we will perform data scaling and use PCA for dimension reduction. As for the algorithms, we are going to use all the above:

- Random Forest
- Stochastic Gradient Descent
- K-Nearest Neighbors
- Linear SVC
- Logistic Regression

Finally, we have to mention that we follow the template given from the instructors for the second assignment and also that the document will be enriched with the code used.

Project Preparation - Exploratory Analysis

The first step of this project is to make all the appropriate preparation. To be more specific we download the dataset and save it to a .csv file. First of all we import all the libraries we need.

```
import pandas as pd
import numpy as np
from sklearn import cross_validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.grid_search import GridSearchCV
from sklearn.datasets import fetch_mldata
from sklearn.decomposition import PCA
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

Also, we use pandas to read our csv and we load our mnist_target with labels and mnist_data with the rest data.

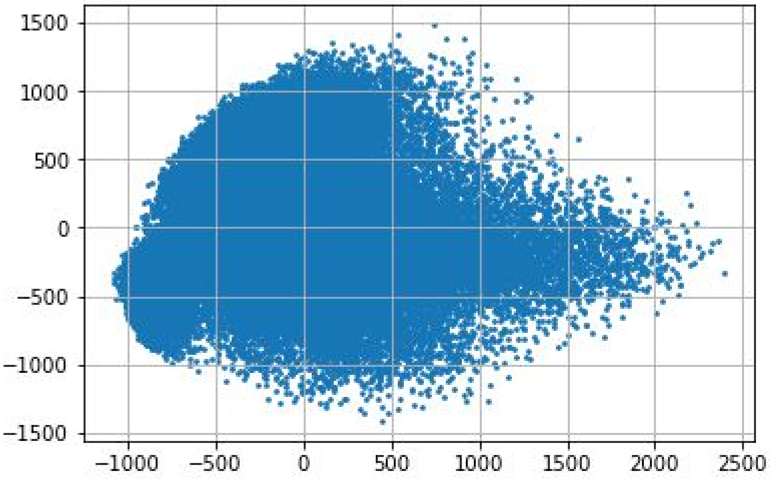
```
mnist = pd.read_csv(‘home/mscuser/Desktop/dataset.csv’)
mnist_target = mnist.label
Mnist_data = mnist.drop([‘label’, axis = 1])
```

In order to find out more about the dataset we can print mnist_data and mnist_target. The results are the expected: (42000, 784) and (42000,)

Data Preparation

We decide not to do any data cleaning. At first, we have decided to perform data wrangling by using StandardScaler, but the result was worst. As a result, we do not perform any data wrangling.

```
x = array_2d[:,0]
y = array_2d[:,1]
plt.scatter(x, y, s=3)
plt.grid(True)
plt.show()
```



Feature Engineering

In order to perform dimension reduction, as mentioned before, we use the PCA.

```
pca = PCA(n_components = 2)
array_2d = pca.fit_transform(mnist_data)
```

Algorithm Selection

To begin with, in order to find out which is the best model for our dataset, we have to split our data to train and test so as to check the accuracy of the algorithms we are going to use to train our model.

```
train = pd.DataFrame(mnist_data)
X = train
y = pd.DataFrame(mnist_target)
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,y,test_size=0.2,random_state=10)]
```

In the next tables we mention the code for each one of the algorithms that have been used and their accuracy.

Random Forest	clf_rf = RandomForestClassifier()
	clf_rf.fit(X_train, (np.array(y_train)).ravel())
	y_pred_rf = clf_rf.predict(X_test)
	acc_rf = accuracy_score(y_test, y_pred_rf)
	print "random forest accuracy: ",acc_rf

SGD	clf_sgd = SGDClassifier()
	clf_sgd.fit(X_train, (np.array(y_train)).ravel())
	y_pred_sgd = clf_sgd.predict(X_test)
	acc_sgd = accuracy_score(y_test, y_pred_sgd)
	print "stochastic gradient descent accuracy: ",acc_sgd

SVC	clf_svm = LinearSVC()
	clf_svm.fit(X_train, (np.array(y_train)).ravel())
	y_pred_svm = clf_svm.predict(X_test)
	acc_svm = accuracy_score(y_test, y_pred_svm)
	print "Linear SVM accuracy: ",acc_svm

K-NN	clf_knn = KNeighborsClassifier()
	clf_knn.fit(X_train, (np.array(y_train)).ravel())
	y_pred_knn = clf_knn.predict(X_test)
	acc_knn = accuracy_score(y_test, y_pred_knn)
	print "nearest neighbors accuracy: ",acc_knn

Logistic Regression	logisticRegr = LogisticRegression()
	logisticRegr.fit(X_train, y_train.ravel())
	y_pred_log = logisticRegr.predict(X_test)
	acc_log = accuracy_score(y_test, y_pred_log)
	print "logistic regression accuracy: ",acc_log

As for the results, you can see the following table

	Random Forest	SGD	K-NN	SVC	Logistic Regression
Accuracy	0,940833	0,865	0,9677	0,86666	0.907738

Considering all the above, we come to a conclusion that the best model is K-NN. in order to find the best parameters for K-NN we will perform param grid.

```
params = {"n_neighbors": np.arange(5,15,30), "metric": ["euclidean", "cityblock", "minkowski"]}

grid = GridSearchCV(clf_knn, params)

grid.fit(X_train, (np.array(y_train)).ravel())

print grid.best_params_
```

The result from the above code shows that the best parameter is euclidean with n_neighbors: 5.

Finally, after taking use of the code we found at github of the class, we use Voting Classifier from the Ensemble methods. In detail this procedure combines Logistic Regression, Decision Tree Classifier, SVC and K-NN.

```
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import KFold
kfold = KFold(n_splits=10, random_state=7)
estimators = []
model1 = LogisticRegression()
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
model4 = KNeighborsClassifier()
estimators.append(('knn', model4))
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, X_train, y_train, cv=kfold)
print(results.mean())]
```

The result (accuracy) of the code above is 0,8650, which does not change our estimation about the best model.