НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Системы искусственного интеллекта

Лабораторная работа № 5

Выполнил студент

Неизвестная Екатерина Павловна

Группа № P33701

Преподаватель: Полещук Елизавета Александровна

г. Санкт-Петербург

2021

Вариант: 1

Задание:

Цель: решить задачу многоклассовой классификации, используя в качестве тренировочного набора данных - набор данных MNIST, содержащий образы рукописных цифр.

1. Используйте метод главных компонент для набора данных MNIST (train dataset объема 60000). Определите, какое минимальное количество главных компонент необходимо использовать, чтобы доля объясненной дисперсии превышала 0.80 + номер_в_списке % 10. Построить график зависимости доли объясненной дисперсии от количества используемых ГК

2. Выведите количество верно классифицированных объектов класса номер_в_списке % 9 для тестовых данных

3. Введите вероятность отнесения 5 любых изображений из тестового набора к назначенному классу

4. Определите Accuracy, Precision, Recall и F1 для обученной модели

5. Сделайте вывод про обученную модель

Отчет:

1)

Код:

```
!pip install --upgrade pip
!pip install --upgrade scikit-learn==0.23.0
```

```
exp_disp = 0.8 + 5 % 10 / 100
classa = 5 % 9
```

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.multiclass import OneVsRestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
from keras.datasets import mnist

(X_train, y_train), (X_pred, y_pred) = mnist.load_data()

dim = 784 # 28*28
X_train_ = X_train.reshape(len(X_train), dim)

pca = PCA(svd_solver='full')
pca = pca.fit(X_train_)

explained_variance = np.round(np.cumsum(pca.explained_variance_ratio_),3)
plt.plot(np.arange(dim), explained_variance, ls = '-')
```

```
M = 0
for arg, val in enumerate(np.cumsum(pca.explained_variance_ratio_)):
  if val > exp_disp:
    M = arg + 1
    break

print("Количество главных компонент, чтобы доля объяснённой дисперсии превышала " + str(exp_disp) + ": " + str(M))
```

```
X_train = X_train.reshape(len(X_train), dim)
pca = PCA(n_components=M, svd_solver='full')
pca = pca.fit(X_train)

explained_variance = np.round(np.cumsum(pca.explained_variance_ratio_),3)
plt.plot(np.arange(M), explained_variance, ls = '-')
```

Выходные данные:

График зависимости доли объясненной дисперсии от всех ГК

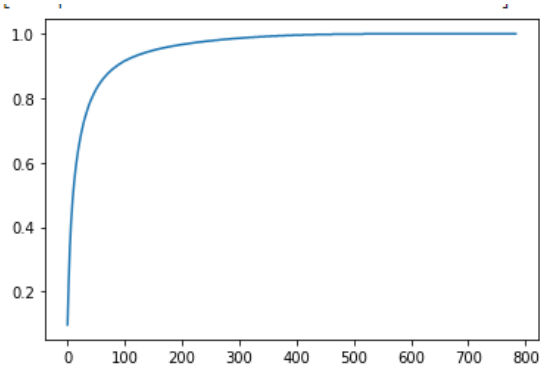
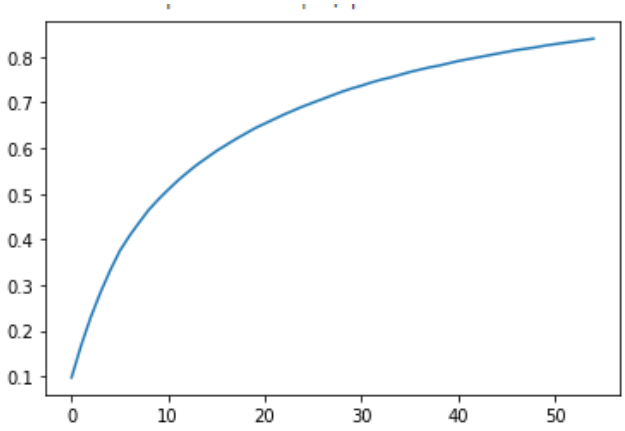


График зависимости доли объясненной дисперсии от количества используемых ГК



2)

Код:

```
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.3, random_state=2020)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)

modelPCA = pca.fit(X_test)
X_test = modelPCA.transform(X_test)

tree = RandomForestClassifier(criterion='gini', min_samples_leaf=10, max_depth=20, n_estimators=10, random_state=2020)
clf = OneVsRestClassifier(tree).fit(X_train, y_train)
y_pred = clf.predict(X_test)

CM = confusion_matrix(y_test, y_pred)
print("Количество верно классифицированных объектов класса " + str(classa) + ": " +
          str(CM[classa][classa]))
```

Выходные данные:

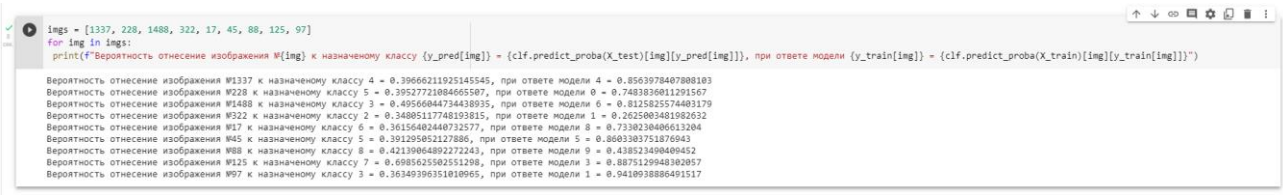
```
Количество главных компонент, чтобы доля объясненной дисперс
Количество верно классифицированных объектов класса 5: 491
```

3)

Код:

```
imgs = [1337, 228, 1488, 322, 17]
for img in imgs:
  print(f"Вероятность отнесение изображения №{img} к назначеному классу {y_pred[img]} = {clf.predict_proba(X_test)[img][y_pred[img]]}")
```

Выходные данные:



Мы видим, что в данном случае модель правильно определила отношение к классу 4 и к классу 5.

Я взяла лишь выборочные изображения, и, чтобы проследить какой из классов угадывается чаще всего, нам нужно провести анализ всех изображений (у нас их очень много, поэтому рассмотрим лишь принцип).

Здесь выводится принадлежность к классу истинная и предсказанная моделью.

4)

Код:

```
from sklearn.metrics import classification_report, accuracy_score
target_names = ['class 0', 'class 1','class 2','class 3','class 4','class 5','class 6','class 7','class 8','class 9']
print("Accuracy:", accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred, target_names=target_names))
```

Выходные данные:

```
Accuracy: 0.5766111111111111
              precision    recall  f1-score   support

     class 0       0.75      0.81      0.78      1693
     class 1       0.91      0.80      0.85      2075
     class 2       0.36      0.50      0.42      1763
     class 3       0.64      0.78      0.70      1873
     class 4       0.62      0.71      0.66      1756
     class 5       0.37      0.31      0.34      1591
     class 6       0.35      0.25      0.29      1766
     class 7       0.74      0.75      0.74      1886
     class 8       0.39      0.36      0.38      1773
     class 9       0.57      0.42      0.48      1824

    accuracy                           0.58     18000
   macro avg       0.57      0.57      0.56     18000
weighted avg       0.58      0.58      0.57     18000
```

Исходя из этой таблицы, мы можем определить, у каких классов всё же чаще выводились правильные ответы. Мы видим, что это классы 0, 1, 3, 4, 7.

Вывод:

В данной лабораторной работе в Google Collab мы обучили модель для предсказания 55-ти нарисованных цифр, чтобы доля объясненной дисперсии была 0,85 на тестовой выборке. У модели получилась точность 0,57.