1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности
5. Высшая школа кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «ТРАДИЦИОННЫЕ МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ»
2. по дисциплине «Основы искусственного интеллекта»
3. Выполнил
4. студент гр. 5151003/00801 В.А. Васильев

<*подпись*>

1. Преподаватель
2. В.М. Крундышев

<*подпись*>

1. Санкт-Петербург
2. 2023

# Теоретические сведения

**Машинное обучение** (англ. machine learning, ML) – это одно из направлений разработки искусственного интеллекта, основанное на выполнении компьютером множества задач без использования прямых инструкций. Ключевое отличие от традиционного программирования заключается в том, что разработчик не задает алгоритм, по которому работает программа, а компьютер сам выбирает методы решения поставленных задач и учится на собственных ошибках.

Машинное обучение базируется на трех основных понятиях:

− **алгоритмы:** специальные программы, «подсказывающие» компьютеру, каким источником данных необходимо воспользоваться. Для каждой задачи подбираются отдельные алгоритмы, составленные с расчетом на ускорение обработки данных и получение точного результата;

− **наборы данных (датасеты, datasets):** информация в виде текстовых, графических, видеофайлов, которую машина использует для накопления опыта при обучении. При этом для решения каждого конкретного типа задач в систему должны загружаться уникальные данные;

− **признаки (свойства, метрики, фичи, features):** индивидуальные измеримые параметры наблюдаемых явлений, от правильности подбора которых зависит точность и скорость работы вычислительной модели.

**Виды машинного обучения:**

− **обучение с учителем (supervised learning):** вычислительная модель обучается на примерах с заранее известными правильными ответами. На основе этих входных примеров и известных правильных ответов требуется восстановить зависимость между множеством примеров и множеством ответов, т.е. построить алгоритм, который будет выдавать достаточно точный ответ для любого примера. Совокупность примеров (входных объектов) и соответствующих им правильных ответов называется обучающей выборкой.

− **обучение без учителя (unsupervised learning):** вычислительная модель обучается на примерах без заранее известных правильных ответов. Система сама находит внутренние взаимосвязи, зависимости, закономерности, существующие между объектами без вмешательства внешнего учителя, экспериментатора, человека.

Комбинированные виды обучения применяют различные сочетания обоих типов обучения в одной программе. Например, обучение с частичным привлечением учителя, обучение с подкреплением и некоторые другие. При обучении с подкреплением учителем является сама окружающая среда, модель среды или неявный учитель.

# Формулировка задания

Цель работы – изучение основных принципов реализации вычислительных моделей на базе машинного обучения и искусственного интеллекта.

# Ход работы

## Наивный Байес (Naive Bayes)

Наивный Байес – это популярный и простой метод машинного обучения, используемый для классификации и вероятностного моделирования. Он основан на теореме Байеса и особенно подходит для классификации текста и фильтрации спама.

С другой стороны, хотя Наивный Байес известен как хороший классификатор, он, как известно, является плохим оценщиком, поэтому к вероятностным выводам predict\_proba не стоит относиться слишком серьезно.

A white background with black text

Description automatically generated

Рисунок 1 – Формула Байеса

**Графическое описание**

Одним из чрезвычайно быстрых способов создания простой модели является предположение о том, что данные описываются гауссовым распределением без ковариации между измерениями. Для этого достаточно найти среднее значение и стандартное отклонение точек в каждой метке, что и требуется для определения такого распределения. Результат такого наивного гауссова предположения показан на следующем рисунке:

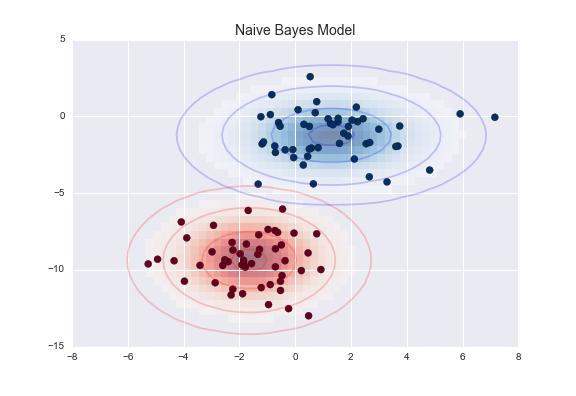


Рисунок 2 – Модель гауссовского наивного Байеса

Эллипсы здесь представляют собой гауссову генеративную модель для каждой метки, причем к центру эллипсов вероятность больше. Имея такую генеративную модель для каждого класса, мы имеем простой рецепт вычисления правдоподобия P(features | L1) для любой точки данных, а значит, можем быстро вычислить апостериорное отношение и определить, какая метка наиболее вероятна для данной точки.

**Формальное описание**

Предполагается, что непрерывные значения всех характеристик имеют распределение Гаусса (нормальное распределение). При нанесении на график получается колоколообразная кривая, симметричная относительно средней значений характеристик, как это показано на рисунке ниже:

A diagram of a gaussian distribution

Description automatically generated

Рисунок 3 – Представление гауссовского наивного Байеса

Предполагается, что это гауссова правдоподобность характеристик, поэтому условная вероятность будет определяться следующим образом:

A black square root equation

Description automatically generated with medium confidence

Рисунок 4 – Условная вероятность по гауссовой правдоподобности

Если в качестве характеристик будут числовые значения, то перед тем как создавать частотные таблицы, нужно преобразовывать числовые значения в их категориальные аналоги. Другим решением может быть использование распределения числовой переменной с приблизительной оценкой частотности, близкой к истинной. Например, один из распространённых методов заключается в применении нормальных распределений или распределений Гаусса для числовых переменных.

Плотность распределения вероятностей для нормального распределения определяется двумя параметрами (среднее значение и среднеквадратическое отклонение) по формуле:

A math equations and formulas

Description automatically generated with medium confidence

Рисунок 5 – Параметры нормального распределения

Наивный байесовский классификатор (Naive Bayes) имеет свои преимущества, но также сопряжен с некоторыми недостатками:

1. Предположение о независимости: одним из основных недостатков является "наивное" предположение о независимости признаков. В реальных данных признаки часто коррелируют между собой, и это предположение может быть сильно нарушено, что может привести к менее точным прогнозам.
2. Чувствительность к качеству данных: Naive Bayes может быть чувствителен к качеству обучающих данных. Например, если в обучающей выборке отсутствуют определенные комбинации признаков, модель не сможет их учесть при классификации.
3. Ограниченность в обработке контекста: из-за предположения о независимости признаков, Naive Bayes не учитывает контекст и порядок признаков, что может быть важным в некоторых задачах, например, в анализе текста.
4. Проблема нулевых вероятностей: если в обучающих данных отсутствуют определенные значения признаков для определенных классов, то Naive Bayes может присваивать им вероятность нуль, что может привести к неправильным прогнозам.
5. Не способен моделировать сложные зависимости: Naive Bayes — это простая модель, которая не способна улавливать сложные и нелинейные зависимости между признаками. В некоторых задачах более сложные модели, такие как нейронные сети, могут быть более эффективными.
6. Низкая точность при малом объеме данных: когда данных мало, Naive Bayes может показывать низкую точность из-за ограниченности статистических оценок вероятностей.

Несмотря на эти недостатки, наивный байесовский классификатор остается полезным инструментом, особенно в задачах классификации текстовых данных и в случаях, когда требуется быстрая и простая модель для базовых задач классификации.

## Optical recognition of handwritten digits dataset

Это набор данных, предоставляемый библиотекой scikit-learn, который содержит изображения рукописных цифр от 0 до 9.

Набор состоит из 8x8 пиксельных изображений рукописных цифр (0-9). Каждое изображение представлено в виде матрицы, где каждый пиксель представляет собой яркость от 0 до 16. Общее количество изображений - 1797.

Каждое изображение представлено как вектор из 64 признаков, где каждый признак соответствует яркости одного пикселя. Таким образом, каждое изображение имеет 64 признака. А также ассоциировано с меткой, которая представляет собой цифру от 0 до 9 и указывает, какую цифру представляет соответствующее изображение.

Этот набор данных часто используется для задач машинного обучения, таких как классификация и кластеризация, а также для демонстрации алгоритмов обработки изображений и распознавания образов в рамках обучения и исследований в области машинного обучения.

## Классификатор

В соответствии с вариантом задания был разработан классификатор на языке Python.

Для реализации гауссовского наивного Байеса был использован соответсвующий метод из библиотеки **sklearn**.

from sklearn.naive\_bayes import GaussianNB  
gnb = GaussianNB()  
y\_pred = gnb.fit(X\_train, y\_train).predict(X\_test)  
print("Number of mislabeled points out of a total %d points : %d" % (X\_test.shape[0], (y\_test != y\_pred).sum()))

Результаты работы программы представлен ниже.

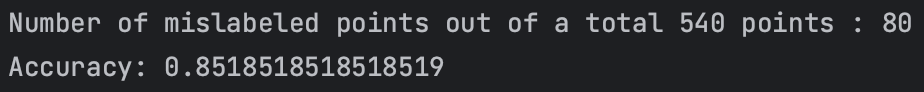


Рисунок 6 – Результат работы классификатора

Производительность вычислительной модели составила 85%, что является хорошим результатом.

## Задача уменьшения размерности

Также была разработана программа, реализующая уменьшение размерности набора данных.

Согласно Методу главных компонент необходимо реализовать проекцию 64-мерных данных о распознанных цифрах в 2-мерные главные компоненты.

A screenshot of a computer screen

Description automatically generated

Рисунок 7 – Результат работы программы, осуществляющей уменьшение размерности

Данный вывод говорит о том, что первая главная компонента содержит 12% информации, а вторая главная компонента содержит только 9% информации. Следует отметить, что при преобразовании четырехмерных данных в двумерные было потеряно целых 79% информации.



Рисунок 8 – Визуализация распределения

Как и было сказано в описании базы данных один класс линейно отделим от двух других, а последние линейно не отделимы друг от друга.

# Выводы

В ходе работы удалось познакомиться с машинным обучением и библиотеками, необходимыми для его реализации. Кроме того, для работы необходимо было изучить метод Наивного Байеса для задачи классификации.

Был создан классификатор на языке Python для работы с базой данных рукописных цифр.

Разработана программа для уменьшения размерности методом главных компонент, в результате было в 32 уменьшено число параметров, но с потерей информации в 79%.

# Приложение А

Классификатор

#Import scikit-learn dataset library  
from sklearn import datasets  
#Load dataset  
digits = datasets.load\_digits()  
  
# print the names of the features  
print("Features: ", digits.feature\_names)  
# print the label type of digits  
print("Labels: ", digits.target\_names)  
  
# print data(feature)shape  
digits.data.shape  
  
# print the digit data features (top 5 records)  
print(digits.data[0:5])  
  
# print the digit labels  
print(digits.target)  
  
# Import train\_test\_split function  
from sklearn.model\_selection import train\_test\_split  
# Split dataset into training set and test set  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(digits.data, digits.target, test\_size=0.3,random\_state=109) # 70% training and 30% test  
  
#Import NB model  
from sklearn.naive\_bayes import GaussianNB  
gnb = GaussianNB()  
y\_pred = gnb.fit(X\_train, y\_train).predict(X\_test)  
print("Number of mislabeled points out of a total %d points : %d" % (X\_test.shape[0], (y\_test != y\_pred).sum()))  
  
# #Import svm model  
# from sklearn import svm  
# #Create a svm Classifier  
# clf = svm.SVC(kernel='linear') # Linear Kernel  
# #Train the model using the training sets  
# clf.fit(X\_train, y\_train)  
# #Predict the response for test dataset  
# y\_pred = clf.predict(X\_test)  
  
#Import scikit-learn metrics module for accuracy calculation  
from sklearn import metrics  
# Model Accuracy: how often is the classifier correct?  
print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

# Приложение Б

Уменьшение размерности

from sklearn.datasets import load\_digits  
  
digits = load\_digits()  
digits\_data = digits.data  
digits\_labels = digits.target  
  
import numpy as np  
  
labels = np.reshape(digits\_labels,(1797,1))  
final\_digits\_data = np.concatenate([digits\_data,labels],axis=1)  
  
import pandas as pd  
  
digits\_dataset = pd.DataFrame(final\_digits\_data)  
features = digits.feature\_names  
features\_labels = np.append(features,'label')  
digits\_dataset.columns = features\_labels  
  
print(digits\_dataset.head())  
  
digits\_dataset['label'].replace(0, 'Zero',inplace=True)  
digits\_dataset['label'].replace(1, 'One',inplace=True)  
digits\_dataset['label'].replace(2, 'Two',inplace=True)  
digits\_dataset['label'].replace(3, 'Three',inplace=True)  
digits\_dataset['label'].replace(4, 'Four',inplace=True)  
digits\_dataset['label'].replace(5, 'Five',inplace=True)  
digits\_dataset['label'].replace(6, 'Six',inplace=True)  
digits\_dataset['label'].replace(7, 'Seven',inplace=True)  
digits\_dataset['label'].replace(8, 'Eight',inplace=True)  
digits\_dataset['label'].replace(9, 'Nine',inplace=True)  
  
print(digits\_dataset.tail())  
  
from sklearn.preprocessing import StandardScaler  
x = digits\_dataset.loc[:, features].values  
x = StandardScaler().fit\_transform(x) # normalizing the features  
  
print(x.shape)  
  
feat\_cols = ['feature'+str(i) for i in range(x.shape[1])]  
normalised\_digits = pd.DataFrame(x,columns=feat\_cols)  
  
print(normalised\_digits.tail())  
  
from sklearn.decomposition import PCA  
pca\_digits = PCA(n\_components=2)  
principalComponents\_digits = pca\_digits.fit\_transform(x)  
principal\_digits\_Df = pd.DataFrame(data = principalComponents\_digits , columns = ['principal component 1', 'principal component 2'])  
  
print(principal\_digits\_Df.tail())  
  
print('Explained variation per principal component: {}'.format(pca\_digits.explained\_variance\_ratio\_))  
  
import matplotlib.pyplot as plt  
plt.figure()  
plt.figure(figsize=(10,10))  
plt.xticks(fontsize=12)  
plt.yticks(fontsize=14)  
plt.xlabel('Principal Component - 1',fontsize=20)  
plt.ylabel('Principal Component - 2',fontsize=20)  
plt.title("Principal Component Analysis of Handwritten Digits Dataset",fontsize=20)  
targets = ['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine']  
colors = ['red', 'green', 'black', 'yellow', 'white', 'brown', 'blue', 'purple', 'grey', 'pink']  
for target, color in zip(targets,colors):  
 indicesToKeep = digits\_dataset['label'] == target  
 plt.scatter(principal\_digits\_Df.loc[indicesToKeep, 'principal component 1'], principal\_digits\_Df.loc[indicesToKeep, 'principal component 2'], c = color, s = 50)  
  
plt.legend(targets,prop={'size': 15})  
plt.show()