МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И РОБОТОТЕХНИКИ

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ»

ОТЧЕТ

по лабораторной работе № 2

Дисциплина «Методы машинного обучения»

Тема «Простейшие методы классификации»

Выполнил студент 4 курса

группы 10702419 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гулиев Э. М.

*(подпись)*

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чваньков А. А.

*(подпись)*

Минск 2022

**Цель работы**: изучение наиболее простых методов классификации.

**Теоретическая часть**

Классификация – это отнесение конкретного объекта (реализа­ции), представленного значениями его свойств (признаков), к одному из фиксированного перечня образов (классов) по определённому решающему правилу (алгоритму) в соответствии с поставленной целью.

Отсюда следует, что классификация может осуществляться любой системой (живой или неживой), выполняющей следующие функции: измерение значений признаков, производство вычислений, реализующих решающее правило. При этом перечень образов, информативных признаков и решающие правила либо задаются системе классификации извне, либо формируются самой системой. Вспомогательная, но важная функция систем классификации – оценка риска потерь. Без этой функции невозможно, например, построить оптимальные решающие правила, выбрать наиболее информативную систему признаков, которые используются при классификации, и др

**Практическая часть**

1. **Описание объектов в виде таблицы Excel**

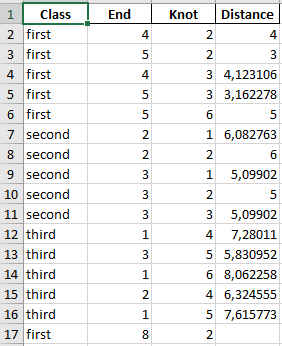
****

Рисунок 1. Представление описания объектов в виде таблицы Excel

1. **Программное обеспечение и результат**

Для написания программы был выбран язык программирования Python. Код программы:

*# Method of "k-nearest neighbors"***import** math  
**import** sys  
**import** tkinter **as** tk  
  
**import** matplotlib.pyplot **as** plt  
**import** pandas **as** pd  
**from** matplotlib.backends.backend\_tkagg **import** FigureCanvasTkAgg  
  
figure = plt.figure(figsize=(6, 4), dpi=125)  
plot = figure.add\_subplot(111)  
featureSpaceClassNameArray = list()  
  
  
**class** Space:  
 objectClass = str  
 knot = **None** end = **None** distance = float  
  
  
**def** IsNaN(string):  
 **return** string != string  
  
  
**def** Distance(objIsTrue, objIsFalse):  
 **return** float(math.sqrt(((objIsFalse.end - objIsTrue.end) \*\* 2) + ((objIsFalse.knot - objIsTrue.knot) \*\* 2)))  
  
  
**def** GetObjectWithoutClass(featureSpaces):  
 objectWithoutClass = Space()  
 index = 0  
 **for** featureSpaceWithoutClass **in** featureSpaces.values:  
 **if** IsNaN(featureSpaceWithoutClass[0]):  
 objectWithoutClass.end = float(featureSpaceWithoutClass[1])  
 objectWithoutClass.knot = float(featureSpaceWithoutClass[2])  
 **break** index += 1  
  
 **if** objectWithoutClass.end **is not None and** objectWithoutClass.knot **is not None**:  
 **return** index, objectWithoutClass  
  
 print(**"Ошибка! Новые точки не найдены!\nПрограмма приостановлена!"**)  
 sys.exit()  
  
  
**def** GetListWithExistingObjectClass(featureSpaces, objectWithoutClass=**None**):  
 objectsWithClass = list()  
 **for** featureSpaceWithClass **in** featureSpaces.values:  
 **if not** IsNaN(featureSpaceWithClass[0]):  
 objectWithClass = Space()  
 objectWithClass.objectClass = featureSpaceWithClass[0]  
 objectWithClass.end = featureSpaceWithClass[1]  
 objectWithClass.knot = featureSpaceWithClass[2]  
 **if** objectWithoutClass **is not None**:  
 objectWithClass.distance = Distance(objectWithClass, objectWithoutClass)  
 objectsWithClass.append(objectWithClass)  
 **return** objectsWithClass  
  
  
**def** GetColorAndSymbolForFeatureSpace(featureSpace):  
 color = **''** symbol = **''  
 if** featureSpace.objectClass == **"first"**:  
 color = **'r'** symbol = **'x'  
 elif** featureSpace.objectClass == **"second"**:  
 color = **'k'** symbol = **'^'  
 elif** featureSpace.objectClass == **"third"**:  
 color = **'b'** symbol = **'\*'  
  
 return** color, symbol  
  
  
**def** GetFeaturesDrawing(featureSpaceWithClassArray):  
 root = tk.Tk()  
 root.resizable(**False**, **False**)  
 root.title(**'Лабораторная работа № 2'**)  
  
 plot.set\_title(**'Простейшие методы классификации'**)  
 plot.set\_xlabel(**'Концевые точки'**)  
 plot.set\_ylabel(**'Узловые точки'**)  
  
 **for** featureSpace **in** featureSpaceWithClassArray:  
 DrawingFeatureSpace(featureSpace)  
  
 **return** root  
  
  
**def** GetWindow(root):  
 plot.grid(**True**)  
 scatter = FigureCanvasTkAgg(figure, root)  
 scatter.get\_tk\_widget().pack(side=tk.LEFT, fill=tk.BOTH)  
 root.mainloop()  
  
  
**def** DrawingFeatureSpace(featureSpace):  
 featureProperties = GetColorAndSymbolForFeatureSpace(featureSpace)  
 color = featureProperties[0]  
 symbol = featureProperties[1]  
 **if** featureSpace.objectClass **in** featureSpaceClassNameArray:  
 label = **""  
 else**:  
 featureSpaceClassNameArray.append(featureSpace.objectClass)  
 label = featureSpace.objectClass  
 plot.scatter(featureSpace.end, featureSpace.knot, **None**, color, marker=symbol, label=label)  
 plot.legend()  
  
  
**def** DrawingLineBetweenFeatureSpace(firstSpaceWithoutClass, secondSpaceWithClass):  
 color = GetColorAndSymbolForFeatureSpace(firstSpaceWithoutClass)[0]  
 plot.plot([firstSpaceWithoutClass.end, secondSpaceWithClass.end],  
 [firstSpaceWithoutClass.knot, secondSpaceWithClass.knot],  
 color=color)  
  
  
**def** EditExcelFile(index, featureExcelData, className, featureSpaceObjects):  
 featureExcelData.loc[index, [**'Class'**]] = className  
 featureExcelData[**'Distance'**] = **None** distanceIndex = 0  
 **for** featureSpaceObject **in** featureSpaceObjects:  
 featureExcelData.loc[distanceIndex, [**'Distance'**]] = featureSpaceObject.distance  
 distanceIndex += 1  
 **with** pd.ExcelWriter(**'Класс признаков.xlsx'**) **as** writer:  
 featureExcelData.to\_excel(writer, index=**False**)  
  
  
**def** main():  
 **if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 featureSpaceFromExcelData = pd.read\_excel(**'Класс признаков.xlsx'**)  
 index, featureSpaceWithoutClass = GetObjectWithoutClass(featureSpaceFromExcelData)  
 featureSpaceWithClassArray = GetListWithExistingObjectClass(featureSpaceFromExcelData, featureSpaceWithoutClass)  
 radius = 2.5  
 nearestFeatureSpace = Space  
  
 **while** nearestFeatureSpace.end **is None and** nearestFeatureSpace.knot **is None**:  
 shortestDistance = radius  
 **for** obj **in** featureSpaceWithClassArray:  
 **if** float(obj.distance) <= radius **and** float(obj.distance) <= shortestDistance:  
 shortestDistance = float(obj.distance)  
 nearestFeatureSpace = obj  
  
 **if** nearestFeatureSpace.end **is None and** nearestFeatureSpace.knot **is None**:  
 print(**"Ни одна из точек не входит в заданный радиус!\nРадиус будет увеличен!"**)  
 radius += 1  
 print(**"Радиус поиска: "** + str(radius))  
 **else**:  
 **break** featureSpaceWithoutClass.objectClass = nearestFeatureSpace.objectClass  
 EditExcelFile(index, featureSpaceFromExcelData, nearestFeatureSpace.objectClass, featureSpaceWithClassArray)  
  
 tkinter = GetFeaturesDrawing(featureSpaceWithClassArray)  
 DrawingFeatureSpace(featureSpaceWithoutClass)  
 DrawingLineBetweenFeatureSpace(featureSpaceWithoutClass, nearestFeatureSpace)  
  
 GetWindow(tkinter)  
  
  
main()

Результат работы программы указан на рисунке 3.

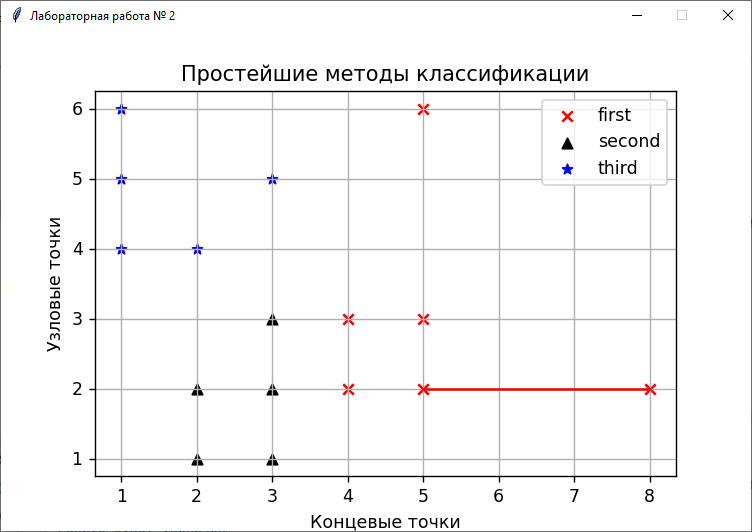
****

Рисунок 2: Результат работы программы.