Гузенко А.М. Группа 7.2. Вариант 4

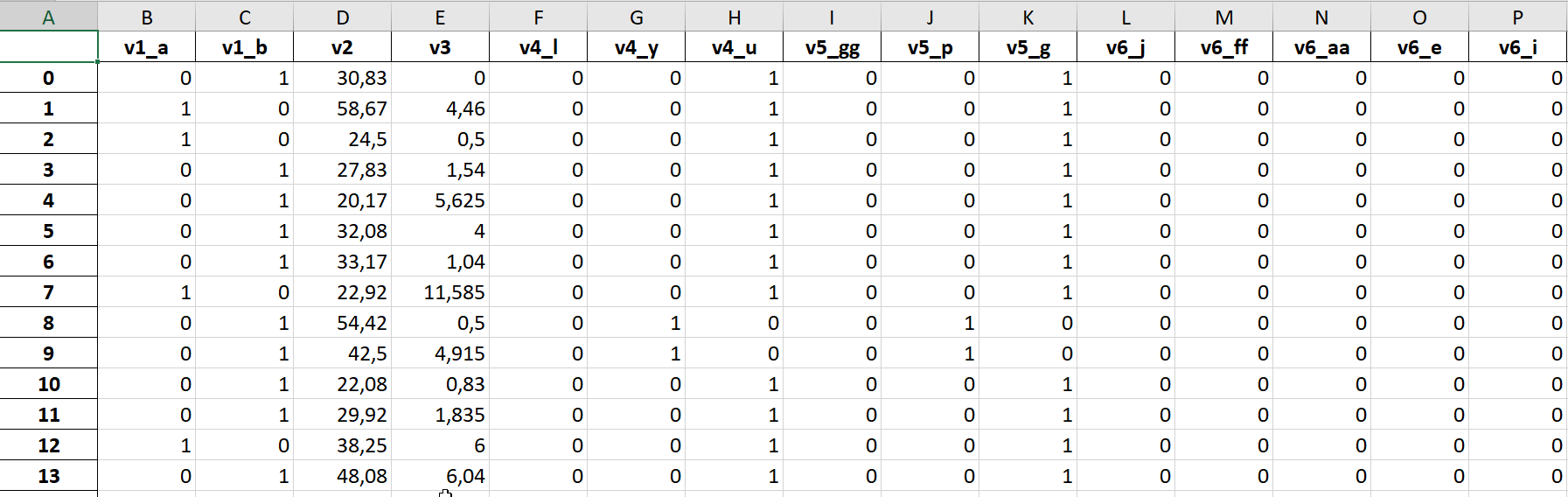
Лабораторная работа № 6

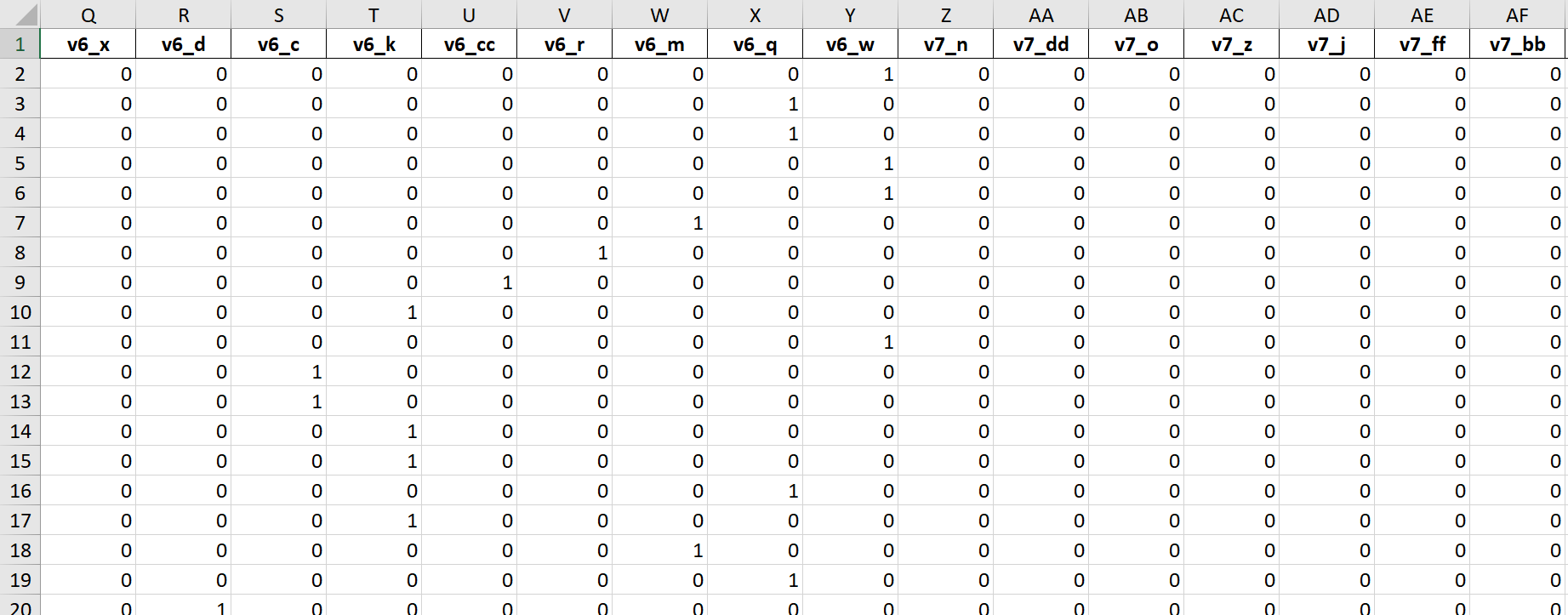
Алгоритм k ближайших соседей

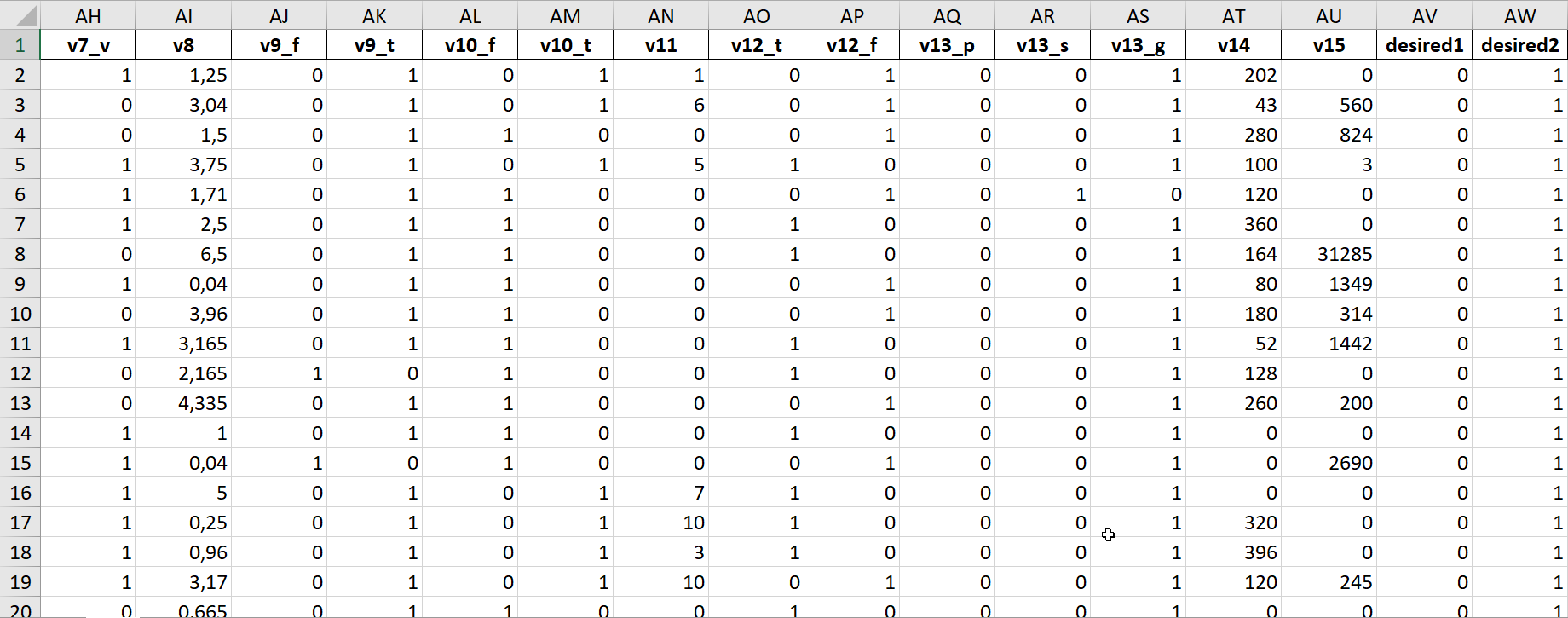
“Принятие решения сотрудниками банка о выдаче кредита”

**Цель**

Применить алгоритм kNN (k ближайших соседей) для предсказания решения сотрудниками банка от неизвестных переменных.

**Описание данных**





Число наблюдений – 655.

Число переменных – 15, из них 6 измерены в количественной (непрерывной) шкале, 9 – в шкале наименований (номинальной шкале).

В файле содержится также два столбца, показывающие, была удовлетворена заявка, или она была отвергнута.

**Выполнение работы**

1. Импортируем нужные библиотеки.

import os  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

1. Объявим константу для пути к программе.

""" CONSTS """  
PATH = os.path.dirname(os.path.abspath(\_\_file\_\_)) + '\\'

1. Прочтем данные из файла

""" CONSTS """  
PATH = os.path.dirname(os.path.abspath(\_\_file\_\_)) + '\\'

1. Вычленим из них нужные для нас переменные.

X = input\_data.iloc[:, 0:46].values  
y = np.array(list(map(lambda x: x[0], input\_data.iloc[:, 47:48].values)))

1. Разделим данные на данные для обучения модели и тестовые.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

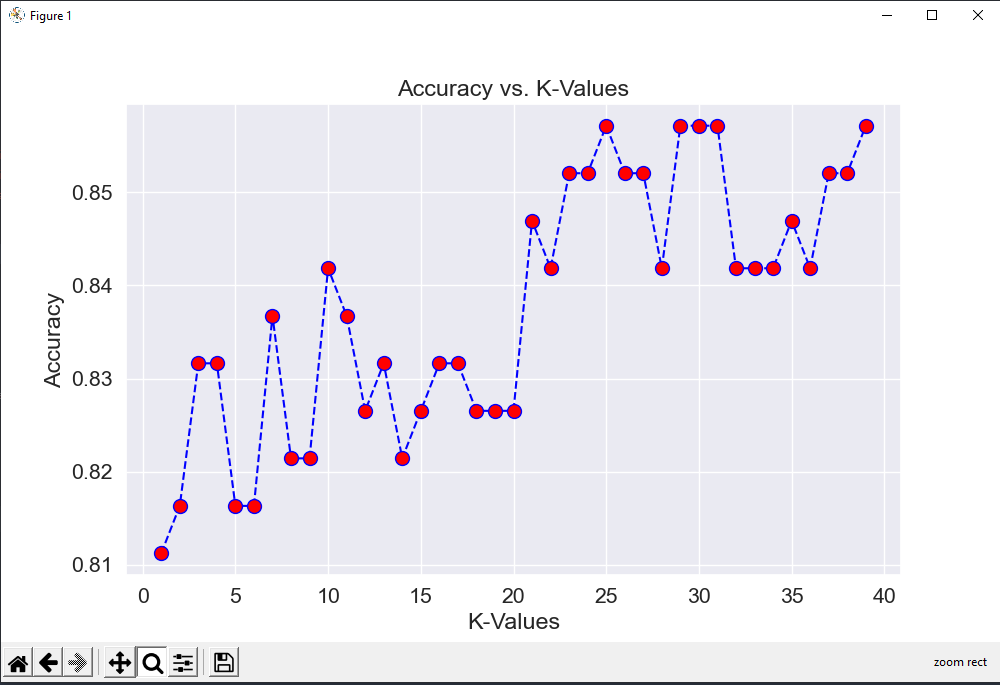
1. Стандартизируем данные путем удаления среднего и масштабирования до единичной дисперсии.

scaler = StandardScaler()  
scaler.fit(X\_train)  
X\_train = scaler.transform(X\_train)  
X\_test = scaler.transform(X\_test)

1. Найдем оптимальное кол-во k.

acc = []

for i in range(1, 40):  
 knn = KNeighborsClassifier(n\_neighbors=i)  
 knn.fit(X\_train, y\_train)  
 acc.append(knn.score(X\_test, y\_test))  
plt.figure(figsize=(10, 4))  
plt.plot(range(1, 40), acc, color='blue', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10)  
plt.title('Accuracy vs. K-Values')  
plt.xlabel('K-Values')  
plt.ylabel('Accuracy')  
plt.show()



1. Возьмем оптимальное кол-во k из предыдущего пункта, создадим модель и обучим ее на данных для обучения.

knn = KNeighborsClassifier(n\_neighbors=39, p=2, metric='minkowski')  
knn.fit(X\_train, y\_train)

1. Сделаем предсказание на тестовых данных.

y\_pred = knn.predict(X\_test)

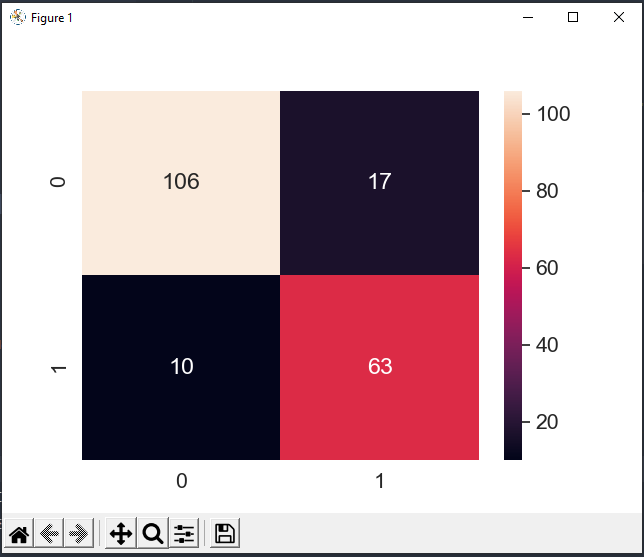
1. Выведем процент ошибки, сравнив предсказанные данные с тестовыми.

print("Accuracy:", accuracy\_score(y\_test, y\_pred))



1. Выведем матрицу ошибок (таблицу сопряженности).

cmat = confusion\_matrix(y\_test, y\_pred)  
sns.set(font\_scale=1.4)  
sns.heatmap(cmat, annot=True, fmt="d")  
plt.show()



**Вывод**

Выполнив данную лабораторную работу, мы создали модель по данным решения сотрудников банка по выдаче кредитов. Мы определили оптимальное кол-во k, равное 39. Модель дает точность в 86%. По матрице ошибок (таблице сопряженности) мы видим, что модель ошибалась в обоих случаях (одобрен/неодобрен кредит) примерно одинаково.