Санкт-Петербургский Национальный Исследовательский

Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Отчет по лабораторной работы 5**

По дисциплине «Системы искусственного интеллекта»

Выполнила:

Мозговая Лариса Андреевна

Группа P33311

Преподаватель:

Кугаевских Александр Владимирович

Санкт-Петербург, 2023

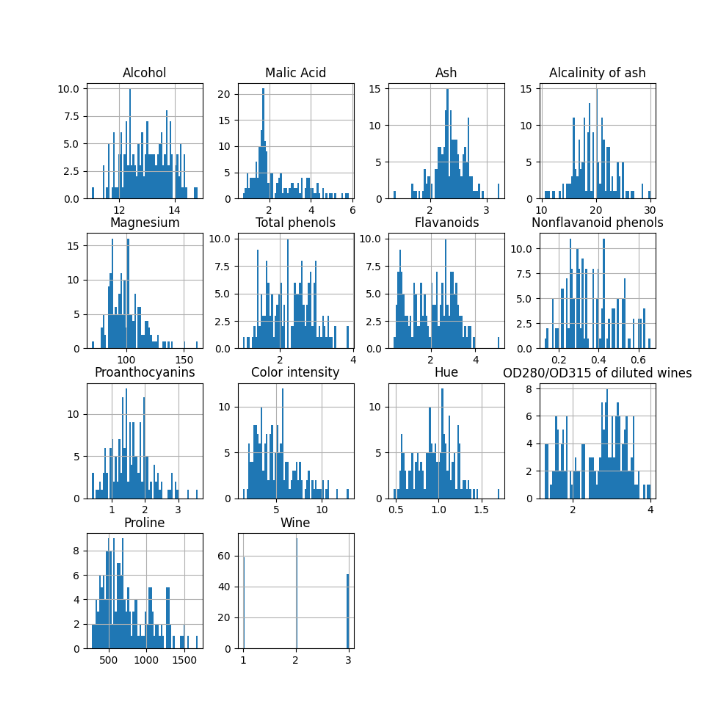
**Задание**

* Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
* Реализуйте метод k-ближайших соседей \*\*\*\*без использования сторонних библиотек, кроме NumPy и Pandas.
* Постройте две модели k-NN с различными наборами признаков:
  + Модель 1: Признаки случайно отбираются.
  + Модель 2: Фиксированный набор признаков, который выбирается заранее.
* Для каждой модели проведите оценку на тестовом наборе данных при разных значениях k. Выберите несколько различных значений k, например, k=3, k=5, k=10, и т. д. Постройте матрицу ошибок.

**Этапы реализации и пояснения:**

Сначала я импортировала нужные библиотеки, далее создала синтаксический признак и вывела статистику по датасету.

data = pd.read\_csv('WineDataset.csv')  
wine\_column = data['Wine']  
#Визуализация статистики по датасету  
data.hist(bins=60, figsize=(10, 10))  
plt.show()



Дальше я сделала обработку отсутствующих значений и маштабирование  
X = data.drop('Wine', axis=1)

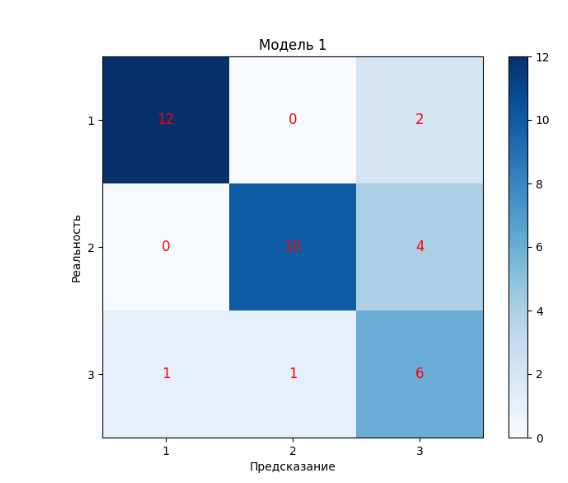
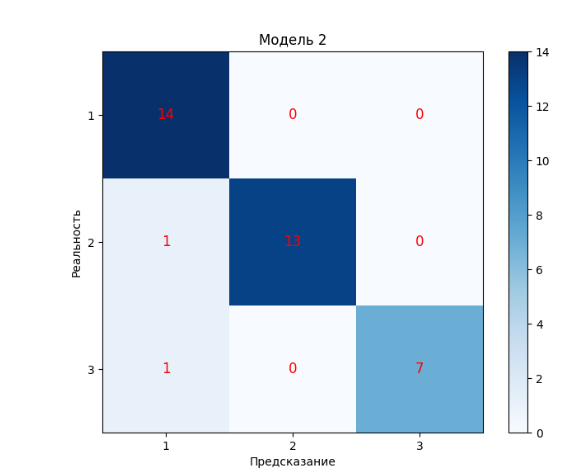
X = (X - X.mean()) / X.std()  
df\_scaled = pd.concat([wine\_column, X], axis=1)  
  
data.fillna(data.mean(), inplace=True)

Далее я разделила данные на обучающий и тестовый набор и реализовала метод k-ближайших соседей def k\_nearest\_neighbors(X\_train, y\_train, x\_test, k, selected\_features):  
 # Евклидово расстояние  
 def euclidean\_distance(x1, x2):  
 return np.sqrt(np.sum((x1[selected\_features] - x2[selected\_features]) \*\* 2))  
  
 # Считаем расстояние  
 distances = [euclidean\_distance(x\_test, x\_train) for x\_train in X\_train]  
 # Берем индексы наблюдений  
 k\_indices = np.argsort(distances)[:k]  
 # По индексам берем метки классов  
 k\_nearest\_labels = [y\_train[i] for i in k\_indices]  
 result\_class = np.bincount(k\_nearest\_labels).argmax()  
 print(result\_class)  
 return result\_class

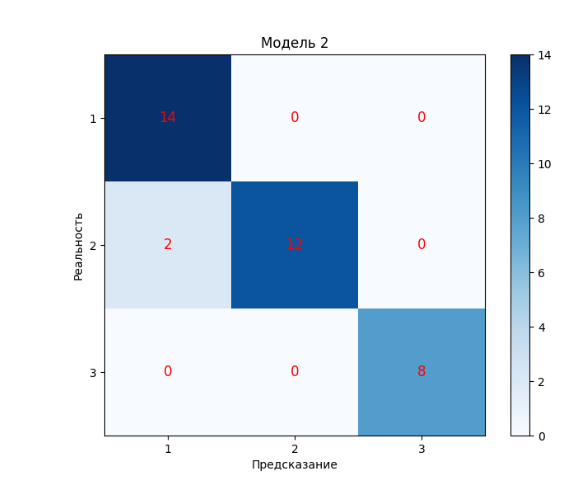
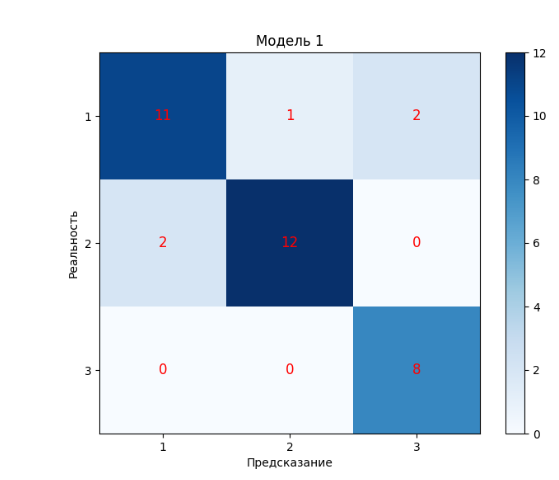
В конце создала 2 модели со случайно отбирающимися признаками и с фиксированным набором признаков

# Создание и обучение Модели 1  
random\_features\_model\_1 = np.random.choice(X\_train.shape[1], size=4, replace=False)  
y\_pred\_model\_1 = []  
for x\_test in X\_test:  
 result\_class = k\_nearest\_neighbors(X\_train, y\_train, x\_test, k=3, selected\_features=random\_features\_model\_1)  
 y\_pred\_model\_1.append(result\_class)  
  
# Создание и обучение Модели 2  
y\_pred\_model\_2 = []  
fixed\_features\_model\_2 = [0, 1, 9, 10, 11]  
for x\_test in X\_test:  
 result\_class = k\_nearest\_neighbors(X\_train, y\_train, x\_test, k=3, selected\_features=fixed\_features\_model\_2)  
 y\_pred\_model\_2.append(result\_class)

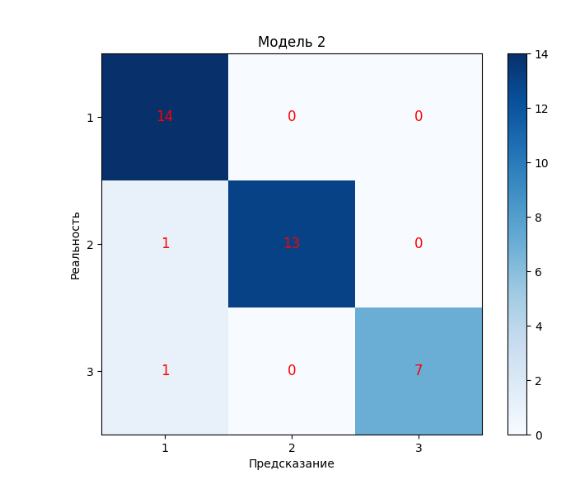
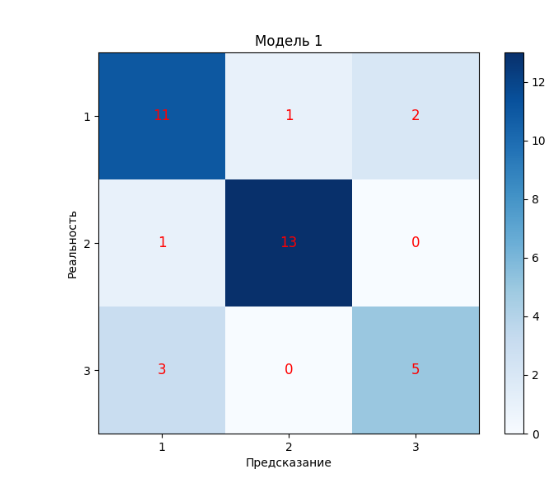
K = 3

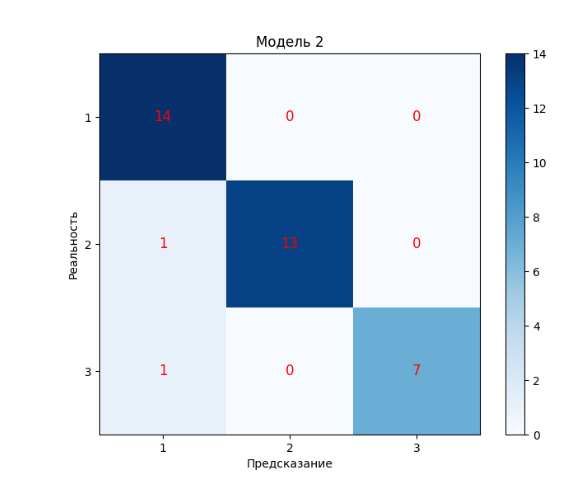
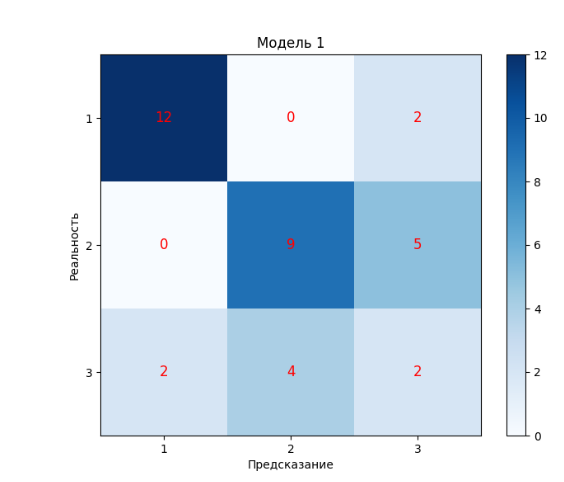
K = 5



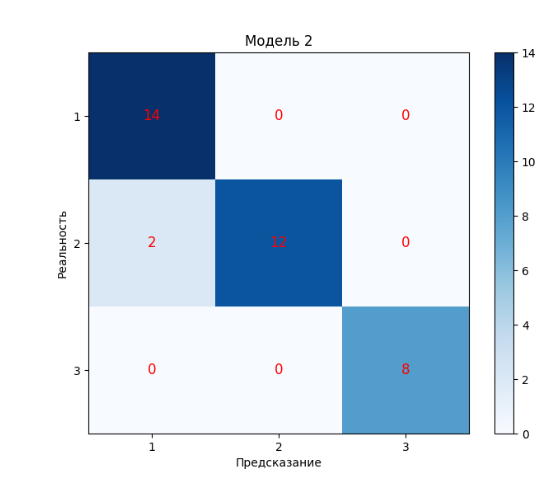
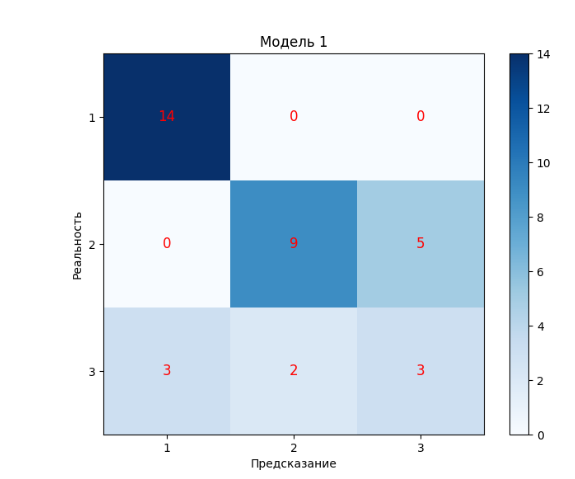
K = 7



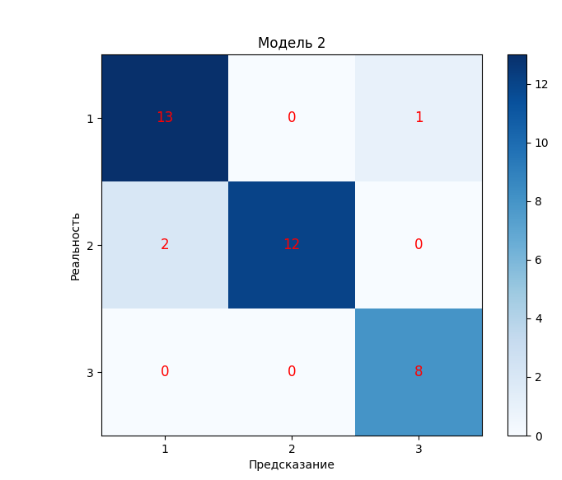
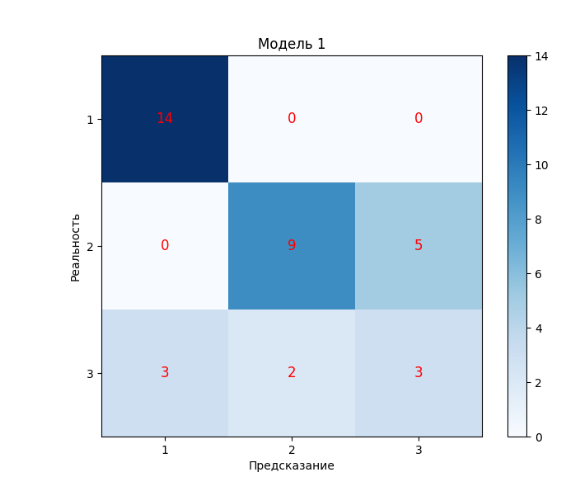
K = 9



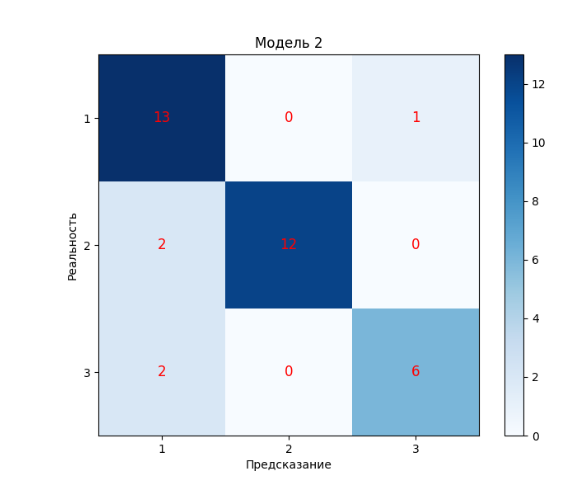
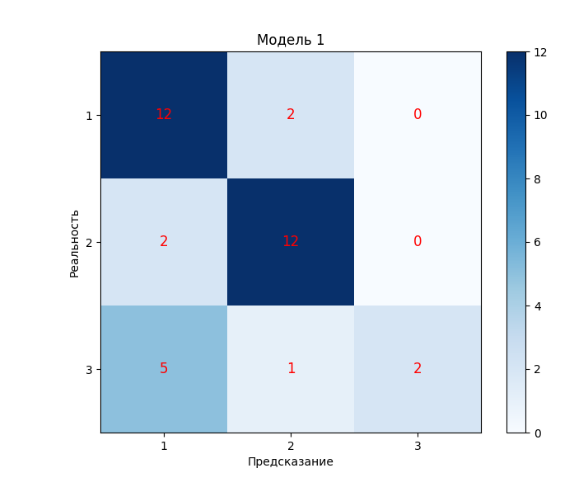
K = 11



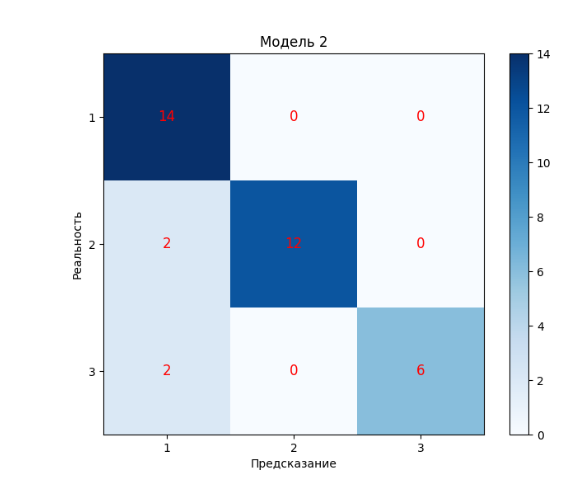
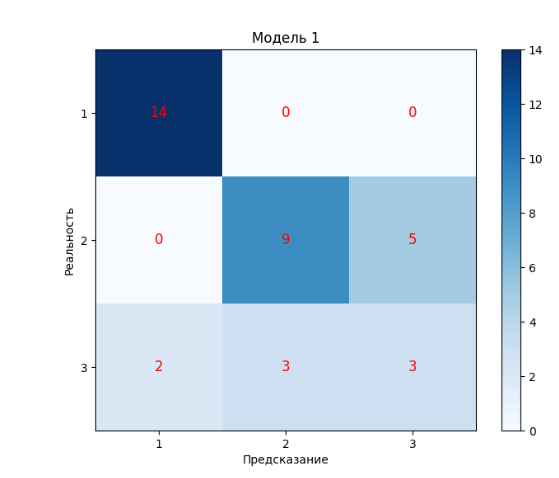
K = 13



K = 15



K = 17



**Вывод:**

При увеличении количества соседей увеличивается погрешность. Оптимальные значения K для этих данных от 3 до 13, в других случаях ошибка выше.