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

**КУРСОВАЯ РАБОТА**

по дисциплине «Машинное обучение, часть1»

1. Выполнил: студент группы
2. 5140201/30301 В.А. Ефременко

*<подпись>*

1. Проверил: Л.В. Уткин
2. д.т.н., профессор
3. *<подпись>*

Санкт-Петербург

2023

1. ***Цель работы***

Курсовой проект заключается в разработке классификаторов для реальной базы данных, визуализации данных, исследовании и настройки классификаторов.

1. ***Формулировка задания***

Для выбранной базы данных необходимо:

1. Разработать 3 классификатора и осуществить настройку их параметров для минимизации ошибки классификации на тестовых данных. Выполнить визуализацию данных при помощи метода t-SNE.
2. Сравнить классификаторы (по критерию вероятность ошибки классификации для тестовых данных) и обосновать выбор наилучшего из них.
3. Удалить их базы метки классов и осуществить кластеризацию данных. Построить дендограмму. Сравнить полученные результаты с реальными метками данных. Определить долю ошибочно кластеризованных данных.
4. Используя логистическую регрессию в рамках метода Лассо, определить наиболее значимые признаки, влияющие на отнесение объектов к определенному классу.
5. Использовать автокодер для сокращения размерности или для реализации разреженного скрытого слоя нейронной сети. Преобразовать обучающую выборку при помощи автокодера и осуществить классификацию новых данных с оценкой ошибки классификации. Выполнить визуализацию новых обучающих данных при помощи метода t-SNE. Определить, когда качество классификации лучше, если использовать сокращение размерности или разреженность скрытого слоя. Выполнить классификацию с использованием зашумленного автокодера (denoising autoencoder). Сравнить полученные результаты с пп.1 и 2.
6. Подготовить пояснительную записку по курсовому проекту и листинги программ.
7. ***Используемые данные и методы***

В качестве датасета была выбрана база данных – Wisconsin Diagnostic Breast Cancer. Набор данных представляет собой результаты анализа биопсийных образцов молочных желез. Задача анализа заключается в диагностике рака молочной железы на основе характеристик ядер клеток, выделенных из дигитализированных изображений тонких игл. Основная информация о датасете:

* Предсказание: B – доброкачественная (benign), M – злокачественная (malignant).
* Множества линейно разделимы с использованием всех 30 входных признаков.
* Признаки вычисляются на основе цифрового изображения тонкой иглы аспирата молочной железы (FNA) и описывают характеристики ядер клеток, присутствующих на изображении.
* Число экземпляров: 569.
* Число атрибутов: 32 (ID, диагноз, вещественнозначные признаки).
* В датасете отсутствуют пропущенные значения.
* Распределение классов: 357 – benign, 212 – malignant.

Для классификации данных были выбраны следующие классификаторы:

* Наивный байесовский классификатор
* Метод ближайших соседей
* Бустинг

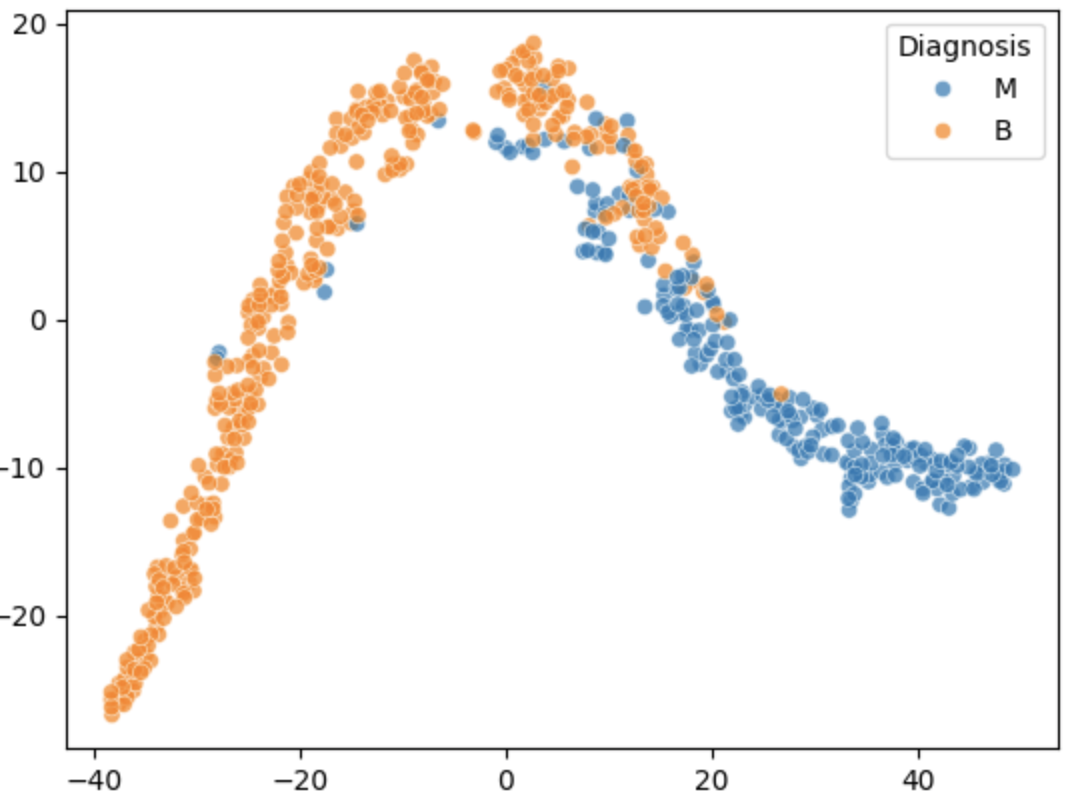
Метод кластеризации: метод k-средних.

1. ***Ход работы***

***Задание 1***

*Разработать 3 классификатора и осуществить настройку их параметров для минимизации ошибки классификации на тестовых данных. Выполнить визуализацию данных при помощи метода t-SNE.*

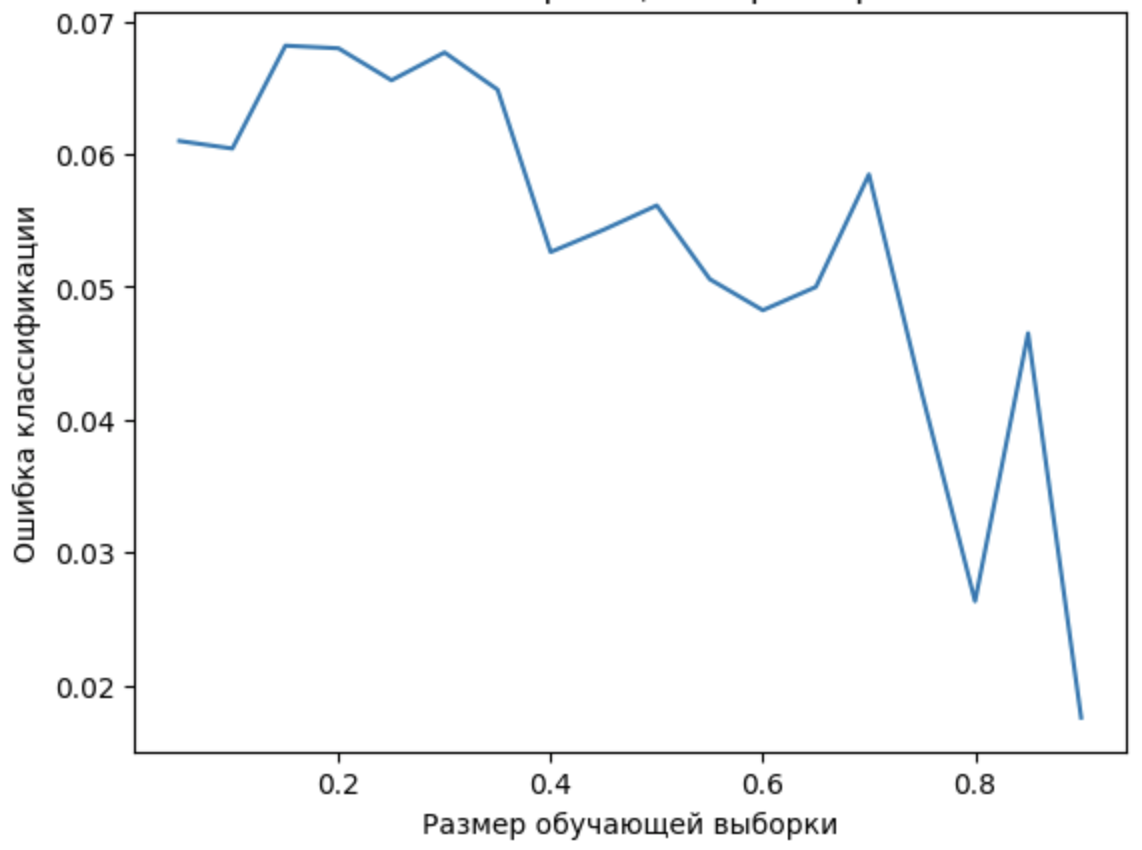
Выполним визуализацию данных при помощи метода t-SNE:



Видим, что классы имеют некоторые пересечения, но при этом визуально они хорошо разделимы.

*Наивный байесовский классификатор*

Для оценки работы байесовского классификатора брались разные доли обучающей выборки от общего объема данных. Была получена следующая зависимость ошибки классификации от размера обучающей выборки:



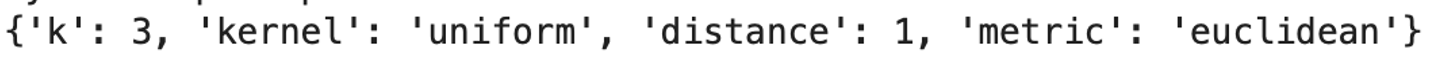
Соответственно, видим, что с увеличением обучающей выборки, ошибка классификации уменьшается и повышается точность классификации.

Наименьшая полученная ошибка классификации: 0.0195.

*Метод ближайших соседей*

Для подбора лучшей модели k-ближайших соседей был проведен эксперимент с разными настройками параметров: k – количества соседей, kernel – ядра, metric – метрики расстояния и distance – расстояния.

Таким образом, наибольшую точность достигла модель с параметрами:

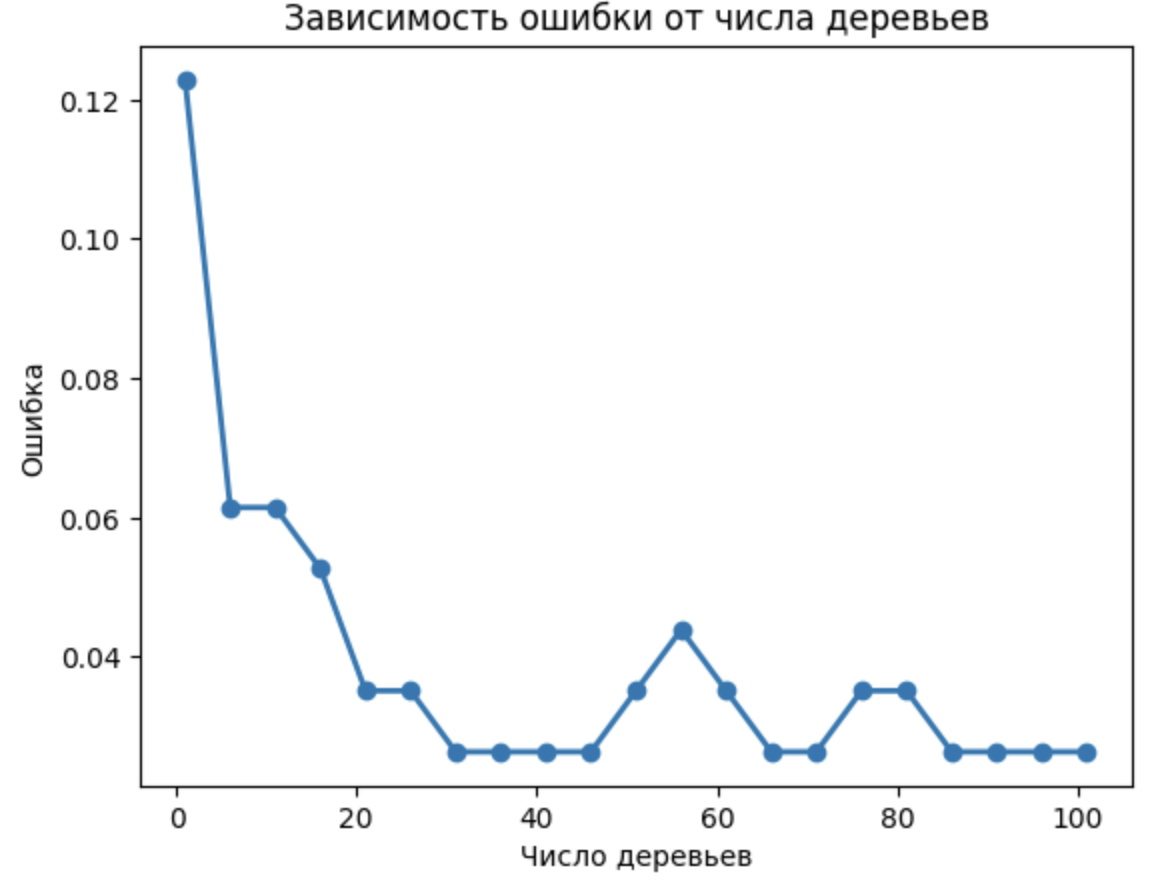


Точность лучшей модели: 0.9211.

*Бустинг*

Для подбора лучшей модели алгоритма бустинга оценивалась ошибка для разного числа деревьев, которое менялось от 1 до 101 с шагом 5. Средняя ошибка моделей была 0.0388.

Наименьшая ошибка была получена для моделей с числом деревьев 31, 36, 41, 46, 66, 71, 86, 91, 96, 101. Наилучшей будем считать модель с числом деревьев 31. Наименьшая полученная ошибка: 0.0263

**

***Задание 2***

*Сравнить классификаторы (по критерию вероятность ошибки классификации для тестовых данных) и обосновать выбор наилучшего из них.*

Ошибки наилучших моделей:

* Наивный байесовский классификатор: 0.0195
* Метод ближайших соседей: 0.0789
* Бустинг: 0.0263

Наилучшим оказался наивный байесовский классификатор, показав наименьшую ошибку классификации на тестовых данных.

***Задание 3***

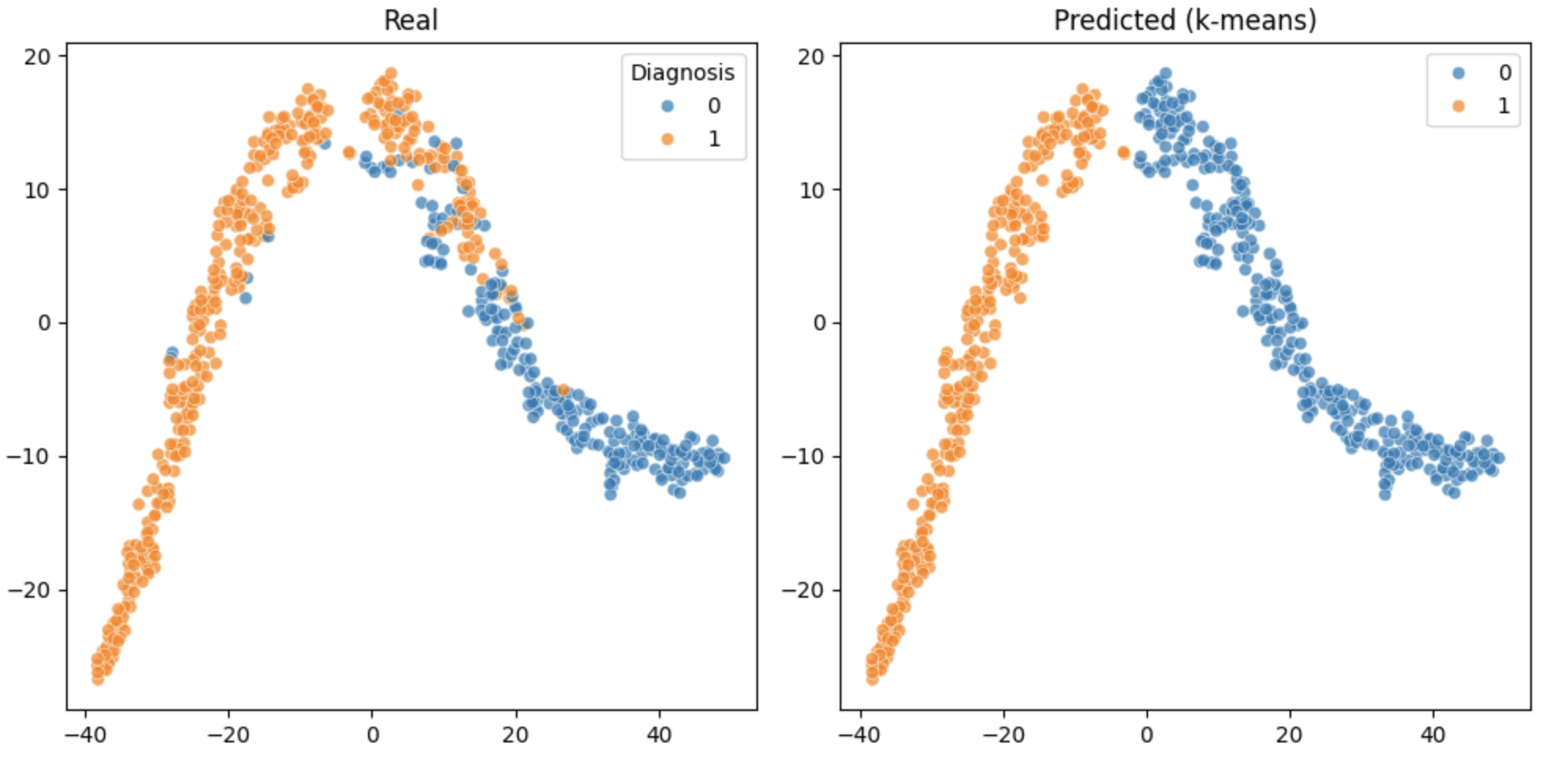
*Удалить из базы метки классов и осуществить кластеризацию данных. Построить дендограмму. Сравнить полученные результаты с реальными метками данных. Определить долю ошибочно кластеризованных данных.*

После удаления меток классов была построена дендрограмма:

*Изображение выглядит как диаграмма, Прямоугольник, снимок экрана, дизайн

Автоматически созданное описание*

Затем была проведена кластеризация данных методом k-средних. Ошибка, полученная с помощью данного метода, составила 17.75%. Ниже представлены графики сравнения данных с исходными метками классов и предсказанными.

**

Видим, что результат довольно неплохой, но модель плохо справляется с пересекающимися участками классов.

***Задание 4***

*Используя логистическую регрессию в рамках метода Лассо, определить наиболее значимые признаки, влияющие на отнесение объектов к определенному классу.*

Определим наиболее значимые признаки и выявим неинформативные, которые лишь мешают работе модели. Сначала используем метод LassoCV для нахождения оптимального параметра alpha = 0.20182966045941297. Затем методом Лассо получим информацию о значимости признаков. Отсортировав признаки по значениям коэффициентов , получили следующую градацию: признаки 24, 23, 5, 25 (именно в таком порядке) наиболее важные, все же остальные признаки не вносят значительного вклада в модель.

***Задание 5***

*Использовать автокодер для сокращения размерности или для реализации разреженного скрытого слоя нейронной сети. Преобразовать обучающую выборку при помощи автокодера и осуществить классификацию новых данных с оценкой ошибки классификации. Выполнить визуализацию новых обучающих данных при помощи метода t-SNE. Определить, когда качество классификации лучше, если использовать сокращение размерности или разреженность скрытого слоя. Выполнить классификацию с использованием зашумленного автокодера (denoising autoencoder). Сравнить полученные результаты с пп.1 и 2.*

**Использование автокодера для сокращения размерности**

Создадим модель автокодера с помощью функции H20DeepLearningEstimator из библиотеки h2о. Чтобы извлечь данные уменьшенной размерности, воспользуемся функцией deepfeatures().

Визуализация данных сокращенной размерности представлена на рисунке ниже:

*Изображение выглядит как снимок экрана, шаблон

Автоматически созданное описание*

Видим, что данные неплохо различимы, но всё еще присутствуют пересечения классов, что повлияет на точность классификации.

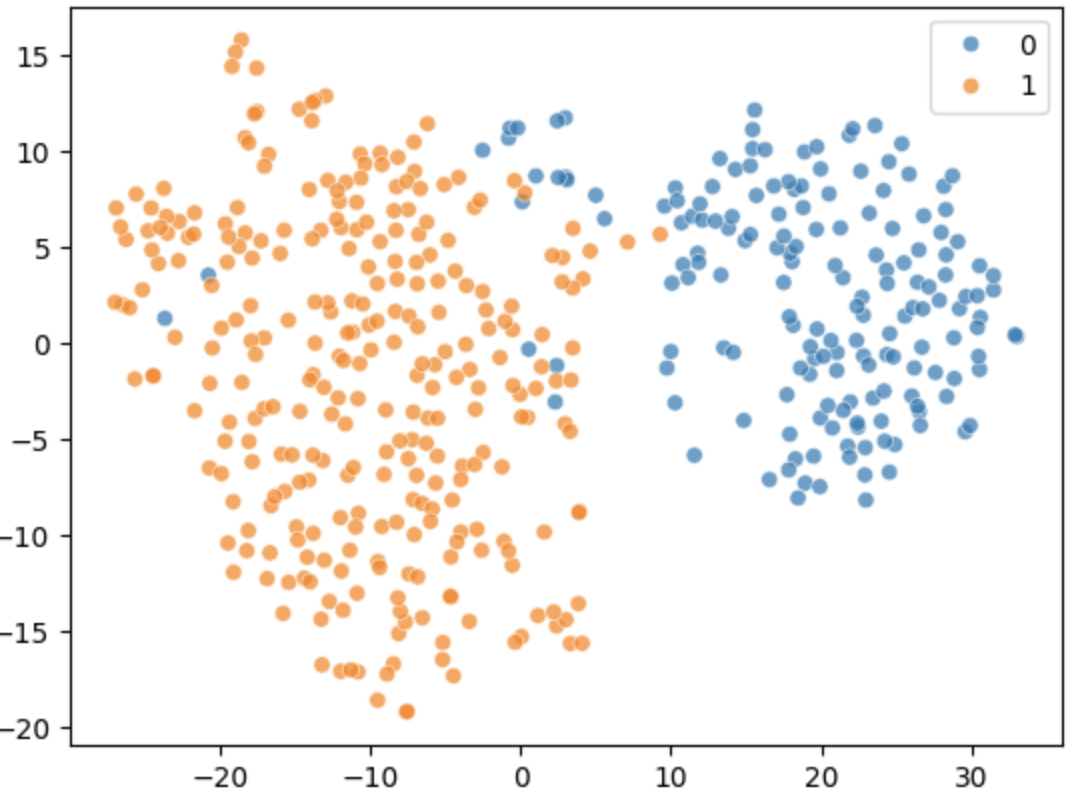
С помощью алгоритма бустинга были получены следующие результаты классификации – ошибка составила 0.0216, что немногим меньше, чем ошибка до уменьшения размерности.

**Использование автокодера для реализации разреженного скрытого слоя**

Автокодер с разреженным скрытым слоем используется для извлечения более информативных и компактных представлений данных. Разреженный скрытый слой в автокодере означает, что только небольшое количество нейронов в этом слое активированы для каждого входного примера, что делает его представление более разреженным.

Разреженные представления позволяют автокодеру извлекать только наиболее важные признаки из данных.

Ниже представлена визуализация данных:

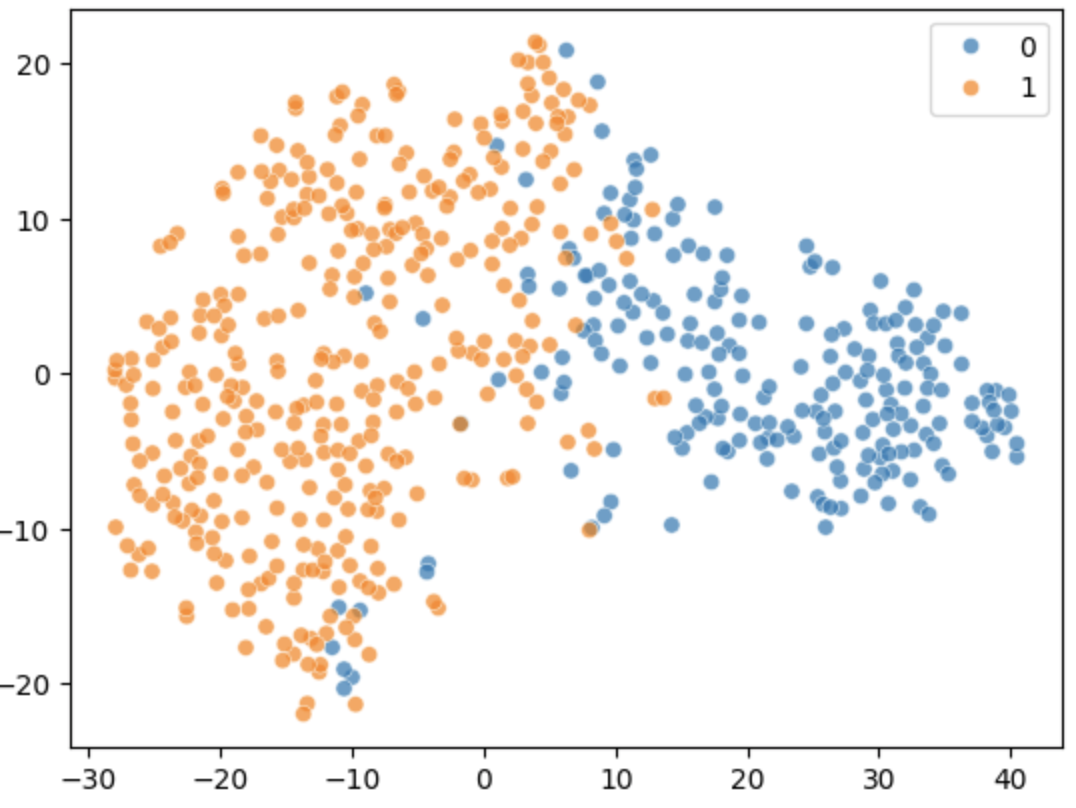
**

С помощью алгоритма бустинга были получены следующие результаты классификации – ошибка составила 0.01075. С использованием разреженности скрытого слоя получили лучшее качество классификации, чем с использованием сокращения размерности.

**Использование зашумленного автокодера**

Шумоподавляющий автокодер (Denoising Autoencoder) — это тип автокодера, который обучается удалять шум из входных данных. Основная цель шумоподавляющего автокодера - изучение более устойчивых и информативных представлений данных путем выделения существенных признаков и подавления влияния шумовых компонентов, поэтому для изучения его влияния предварительно зашумляем исходные данные путем добавления случайного шума.

Ниже представлена визуализация данных после прохождения зашумленным автокодером:



С помощью алгоритма бустинга были получены следующие результаты классификации – ошибка составила 0.0526. Шумоподавляющий автокодер дает самый низкий результат.

Сравнивая полученные результаты, можно сделать вывод, что на данном наборе данных лучше работать с разряженным автокодером.

1. ***Выводы***

В данной работе для базы данных «Wisconsin Diagnostic Breast Cancer»:

1. Реализованы 3 классификатора, такие как наивный байесовский классификатор, k-ближайших соседей и бустинг. Для данных классификаторов были подобраны параметры для получения наименьшей ошибки классификации. Наилучшим классификатором оказался наивный байесовский.
2. Реализован метод кластеризации k-средних. Точность кластеризации составляет 82.25%, что является неплохим результатом.
3. Осуществлена градация признаков, влияющих на отнесение объектов к определенному классу, с помощью логистической регрессии в рамках метода Лассо.
4. Исследовано применение разных автокодеров. В качестве метода классификации новых данных был использован алгоритм бустинга с выявленными ранее оптимальными параметрами. Использование разреженного автокодера дало наименьшую ошибку классификации - 0.01075. Худшую ошибку классификации показало использование шумоподавляющего автокодера.
5. ***Листинг кода***

***Задание 1***

############ Наивный байесовский классификатор

def train\_and\_evaluate(test\_size):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=test\_size, random\_state=42)

nb\_classifier = GaussianNB()

nb\_classifier.fit(X\_train, y\_train)

y\_pred = nb\_classifier.predict(X\_test)

error\_rate = np.mean(y\_pred != y\_test)

return error\_rate, nb\_classifier, X\_test, y\_test, y\_pred

test\_sizes = np.arange(0.1, 1.0, 0.05)

error\_rates = []

best\_model = None

best\_error\_rate = float('inf')

for test\_size in test\_sizes:

error\_rate, model, X\_test, y\_test, y\_pred = train\_and\_evaluate(test\_size)

error\_rates.append(error\_rate)

if error\_rate < best\_error\_rate:

best\_error\_rate = error\_rate

best\_model = model

best\_X\_test = X\_test

best\_y\_test = y\_test

best\_y\_pred = y\_pred

############ Метод ближайших соседей

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

k\_values = list(range(1, 11))

kernel\_values = ["uniform", "distance"]

distance\_values = list(range(1, 11))

metric\_values = ["euclidean", "manhattan", "chebyshev"]

best\_accuracy = 0

best\_params = {}

for k, kernel, distance, metric in product(k\_values, kernel\_values, distance\_values, metric\_values):

model = KNeighborsClassifier(n\_neighbors=k, weights=kernel, p=distance, metric=metric)

model.fit(X\_train, y\_train)

pred = model.predict(X\_test)

accuracy = (pred == y\_test).mean()

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_params = {'k': k, 'kernel': kernel, 'distance': distance, 'metric': metric}

best\_model = KNeighborsClassifier(n\_neighbors=best\_params['k'],

weights=best\_params['kernel'],

p=best\_params['distance'],

metric=best\_params['metric'])

best\_model.fit(X\_train, y\_train)

pred = best\_model.predict(X\_test)

final\_accuracy = (pred == y\_test).mean()

print(f"Точность лучшей модели: {final\_accuracy:.4f}")

print("Лучшие параметры:")

print(best\_params)

############ Бустинг

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

y\_train = y\_train.astype('category')

y\_test = y\_test.astype('category')

tree\_num = np.arange(1, 102, 5)

error\_boost = []

for t in tree\_num:

err\_boost = []

for i in range(5):

clf = AdaBoostClassifier(n\_estimators=t, random\_state=i)

clf.fit(X\_train, y\_train)

pred = clf.predict(X\_test)

err\_boost.append(1 - accuracy\_score(y\_test, pred))

error\_boost.append(np.mean(err\_boost))

print("Средняя ошибка:", np.mean(error\_boost))

print("Ошибки для каждого числа деревьев:", error\_boost)

***Задание 3***

############ Дендрограмма

distance\_matrix = hierarchy.distance.pdist(X\_emb)

linkage\_matrix = hierarchy.linkage(distance\_matrix, method='single')

plt.figure(figsize=(10, 6))

dendrogram = hierarchy.dendrogram(linkage\_matrix, labels=y.values)

plt.show()

############ k-средних

file\_path\_data = 'wdbc.data'

df = pd.read\_csv(file\_path\_data, header=None, delimiter=',')

df.columns = ['ID', 'Diagnosis'] + list(df.columns[2:])

df['Diagnosis'] = df['Diagnosis'].replace({'M': 0, 'B': 1})

X = df.drop(columns=['ID', 'Diagnosis'])

y = df['Diagnosis']

X\_emb = TSNE(n\_components=2).fit\_transform(X)

kmeans = KMeans(n\_clusters=2, random\_state=42)

predict\_kmeans = kmeans.fit\_predict(X\_emb)

errors\_kmeans = (predict\_kmeans != y).sum() / len(y)

print(f'Error (k-means): {errors\_kmeans \* 100:.2f}%')

fig, ax = plt.subplots(1, 2, figsize=(10, 5))

sns.scatterplot(ax=ax[0], x=X\_emb[:, 0], y=X\_emb[:, 1], hue=y, alpha=0.7)

ax[0].set\_title('Real')

sns.scatterplot(ax=ax[1], x=X\_emb[:, 0], y=X\_emb[:, 1], hue=predict\_kmeans, alpha=0.7)

ax[1].set\_title('Predicted (k-means)')

fig.tight\_layout()

plt.show()

***Задание 4***

############ Лассо

lasso\_cv = LassoCV(cv=100, random\_state=25)

lasso\_cv.fit(X, y)

optimal\_alpha = lasso\_cv.alpha\_

print(optimal\_alpha)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=25)

lasso\_model = Lasso(alpha=optimal\_alpha, random\_state=25)

lasso\_model.fit(x\_train, y\_train)

sorted\_features = sorted(zip(lasso\_model.coef\_, X.columns), key=lambda x: abs(x[0]), reverse=True)

for coef, feature\_name in sorted\_features:

print(f'{feature\_name}: {coef}')

***Задание 5***

########## Сокращение размерности автокодером

data = h2o.H2OFrame(df)

data['Diagnosis'] = data['Diagnosis'].asfactor()

train, val = data.split\_frame([0.8])

predictors = [f"{i}" for i in range(2, 32)]

response = "Diagnosis"

mdl = H2ODeepLearningEstimator(activation='tanh', autoencoder=True)

mdl.train(predictors, response, training\_frame=train)

features = mdl.deepfeatures(train, layer=1)

x\_new = features.as\_data\_frame()

y\_new = train[response].as\_data\_frame().values.reshape(1, -1)

emb = TSNE(n\_components=2).fit\_transform(x\_new)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_new, y\_new[0], test\_size=0.8, stratify=y\_new[0], random\_state=25)

Ada\_model = AdaBoostClassifier(n\_estimators=31).fit(x\_train, y\_train)

y\_pred = Ada\_model.predict(x\_test)

print(f'Ошибка: {(y\_pred != y\_test).sum()/len(y\_test)}')

########## разряж автокодер

mdl = H2ODeepLearningEstimator(activation='tanh', autoencoder=True, hidden=[500,500])

mdl.train(predictors, response, training\_frame=train)

features = mdl1.deepfeatures(train, layer=1)

x\_new = features.as\_data\_frame()

emb = TSNE(n\_components=2).fit\_transform(x\_new)

sns.scatterplot(x=emb[:,0], y=emb[:,1], hue=y\_new[0], alpha=0.7)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_new, y\_new[0], test\_size=0.2, stratify=y\_new[0], random\_state=25)

Ada\_model = AdaBoostClassifier(n\_estimators=31).fit(x\_train, y\_train)

y\_pred = Ada\_model.predict(x\_test)

print(f'Error: {(y\_pred != y\_test).sum()/len(y\_test)}')

########## зашумленный автокодер

# Часть данных которая будет зашумлена

subset\_size = 100

subset\_indices = np.random.choice(df.shape[0], size=subset\_size, replace=False)

X = df.drop(columns=['ID', 'Diagnosis'])

y = df['Diagnosis'].tolist()

X\_subset = X.iloc[subset\_indices]

y\_subset = [y[i] for i in subset\_indices]

noise\_subset = np.random.normal(0, 1, size=(subset\_size, X.shape[1]))

X\_subset\_with\_noise = X\_subset + noise\_subset

X\_with\_noise = X.copy()

X\_with\_noise.iloc[subset\_indices] = X\_subset\_with\_noise

train = h2o.H2OFrame(pd.concat([X\_with\_noise, pd.Series(y, name='Diagnosis')], axis=1))

mdl1 = H2ODeepLearningEstimator(activation='tanh', autoencoder=True)

mdl1.train(predictors, response, training\_frame=train, validation\_frame=val)

features = mdl1.deepfeatures(train, layer=1)

a = train['Diagnosis'].as\_data\_frame().values.reshape(1, -1)

data\_new = features.as\_data\_frame()

emb = TSNE(n\_components=2).fit\_transform(data\_new)

sns.scatterplot(x=emb[:, 0], y=emb[:, 1], hue=a[0], alpha=0.7)

plt.show()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data\_new, a[0], test\_size=0.2, stratify=a[0], random\_state=25)

bag\_model = AdaBoostClassifier(n\_estimators=31).fit(x\_train, y\_train)

predict2 = bag\_model.predict(x\_test)

print(f'Error: {(predict2 != y\_test).sum()/len(y\_test)}')