Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский химико-технологический университет имени Д.И. Менделеева»

Факультет цифровых технологий и химического инжиниринга

Кафедра информационных компьютерных технологий

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 14**

**ПО КУРСУ**

**«ЦИФРОВОЕ МОДЕЛИРОВАНИЕ ФИЗИКО-ХИМИЧЕСКИХ СИСТЕМ»:**

**«Предсказание свойств материалов с помощью ML»**

Ведущий преподаватель

Ст. преподаватель Скичко Е.А.

**СТУДЕНТ группы КС-20** Мелехин А.А.

**Москва**

**2024**

# **Задание**

Разработать ML-модель для предсказания плотности воздуха. Число параметров в модели определить из правила. Построить графики зависимости Свойства от Температуры *[K]* для различных значений *P=const [бар]*, на графиках отметить точками экспериментальные значения из БД, линиями – результаты предсказания модели. Предсказать свойство для указанных значений давления и температуры и отметить на тех же графиках цветными точками. Модель сдается в виде кода и сохраняется в файле ML.keras.

**Теоретическое обоснование решения**

**Физические свойства** вещества: вязкость, плотность, температура плавления, температура кипения, диэлектрическая проницаемость, теплоёмкость, теплопроводность, электропроводность, показатель преломления.

**Структурные свойства** вещества: постоянные кристаллической решетки и углы, валентные углы, длины связей.

**Физико-химические** свойства вещества: энтальпия образования, энергия связи, дипольные моменты молекул.

**Machine learning** (машинное обучение) – методы, которые используют обучение на данных для решения сходных задач. Три основных вида:

1) **обучение с учителем** (supervised learning); На вход модели поступают размеченные данные (датасет) – входы и желаемые выходы. Модель сравнивает свой выход с желаемым и меняет свои параметры для улучшения рассогласования (метрика ошибки). Основная решаемая задача – предсказание (регрессия) и классификация (в заданное число классов).

2) **обучение без учителя** (unsupervised learning); Классы не заданы. Желаемого выхода нет. Основная решаемая задача – кластеризация (группировка данных по заранее не заданным кластерам).

3) **обучение с подкреплением** (reinforcement learning) Есть критерий оценки работы в целом (без отдельных примеров). Алгоритм изменяется так, чтобы улучшить критерий.

Для решения задач ML часто используют **ИНС** – набор искусственных нейронов, соединенных синаптическими связями. Имитируют работу нервной системы (передачу сигналов). Существуют множество типов ИНС. Самый простой для решения задачи предсказания – многослойный перцептрон. Нейроны обозначают кругами, которые группируются в слои в направлении прохождения сигналов, а связи – линиями. Стандартный мн. перц. – полносвязанный между соседними слоями.

**Обучение нейросети**

**Обучение нейросети** состоит из фиксированного числа эпох (например, 1000). На каждой эпохе все данные обучающей выборки по одному образцу (строке) поступают на вход ИНС. Корректируются весовые коэффициенты. Они меняются так, чтобы уменьшить функцию потерь (loss function). Эта функция задается обычно равной некой метрике качества предсказания нейросети. По окончании подачи на вход всех строк – повторяем процесс сначала (новая эпоха обучения). После окончания обучения рассчитываются метрики качества предсказания ИНС на обучающей и тестовой выборке.

**Датасет** (строки) делится на две части:

- Обучающая выборка (80%)

- Тестовая выборка (20%)

**Правила:**

1) число связей ИНС должно быть < числа примеров в обучающей выборке из датасета. Традиционно, в 10 раз.

2) число нейронов скрытого слоя

i - входной слой, o – выходной, a = 2 для начала.

**Кроссвалидация** – делим выборку по-разному. 5-кратная кроссвалидация – 5 раз обучаем полностью ИНС с нуля, причем 20% данных для тестовой выборки берем все 5 раз по-разному. Пусть задано 1000 эпох обучения. При 5-кратной кроссвалидации всего будет 5000 эпох с разным разбиением данных и 5 разных итоговых вариантов модели. Выбирается наилучшая модель.

Пусть задано 1000 эпох обучения. При 5-кратной кроссвалидации всего будет 5000 эпох с разным разбиением данных и 5 разных итоговых вариантов модели. Выбирается наилучшая модель.

**Метрика качества предсказания**

Рассчитываются по каждому выходу модели отдельно. Для некоторой выборки (например, тестовой или обучающей) Основные две метрики:

**MAPE** (средняя процентная ошибка = средняя относительная ошибка):

**MAE** (средняя абсолютная ошибка):

- значение выхода из i-ой строки данных.

– значение выхода, рассчитанное ИНС, при поступлении на вход i-ой строки данных.

*n* - число строк данных в выборке.

**Алгоритм выполнения задания**

1. Подготовить файл *data.csv* с входными данными (программа *data.py*)
2. Нормализовать входные данные (функция *normalize\_data* из программы *normalize.py*)
3. Обучить нейросеть (программа *main.py*)
4. Выполнить предсказание на всех данных (программа *predict.py*)
5. Денормализовать данные (функция *denormalize\_data* из программы *normalize.py*)
6. Построить график (программа *predict.py*)

**Код**

**Программа data.py**

import csv

f = open("temp.txt", "r")

T = list(map(float, f.readline().split()))

f = open("pressure.txt", "r")

P = list(map(float, f.readline().split()))

f = open("density.txt", "r")

D = [list(map(float, line.split())) for line in f.readlines()]

f = open("data.txt", "w")

for i in range(len(P)):

    for j in range(len(T)):

        Temperature = str(round(T[j] + 273.15, 2))

        Presure = str(P[i])

        Density = str(D[i][j])

        line = Temperature + "," + Presure + "," + Density + "\n"

        f.write(line)

f.close()

with open('data.txt', 'r') as in\_file:

    stripped = (line.strip() for line in in\_file)

    lines = (line.split(",") for line in stripped if line)

    with open('../data.csv', 'w') as out\_file:

        writer = csv.writer(out\_file)

        writer.writerow(('Temperature', 'Pressure', 'Density'))

        writer.writerows(lines)

**Программа main.py**

from math import log

from math import exp

from numpy import mean

import numpy as np

import pandas as pd

from sklearn.model\_selection import RepeatedKFold

from keras.models import Sequential

from keras.layers import Dense

from keras import optimizers

from matplotlib import pyplot as plt

from normalize import normalize\_data, denormalize\_data

from get\_dataset import get\_dataset

def get\_model(n\_inputs, n\_outputs):

    model = Sequential()

    model.add(Dense(2, input\_dim=n\_inputs, activation='sigmoid'))

    model.add(Dense(n\_outputs, activation='linear'))

    opt1 = optimizers.Adam(learning\_rate=0.005)

    model.compile(loss='mae', metrics=['mape'], optimizer=opt1)

    model.summary()

    return model

def evaluate\_model(X, y):

    n\_inputs, n\_outputs = X.shape[1], y.shape[1]

    cv = RepeatedKFold(n\_splits=5, n\_repeats=1, random\_state=22527)

    i = 0

    MAPE = 300

    for train\_ix, test\_ix in cv.split(X):

        i = i + 1

        X\_train, X\_test = X[train\_ix], X[test\_ix]

        y\_train, y\_test = y[train\_ix], y[test\_ix]

        model = get\_model(n\_inputs, n\_outputs)

        history = model.fit(X\_train, y\_train, verbose=0, epochs=1000)

        plt.plot(history.history['loss'])

        plt.title('model loss')

        plt.ylabel('loss')

        plt.xlabel('epoch')

        plt.legend(['train', 'val'], loc='upper left')

        [mae\_train, mape\_train] = model.evaluate(X\_train, y\_train)

        [mae\_test, mape\_test] = model.evaluate(X\_test, y\_test)

        [mae, mape] = model.evaluate(X, y)

        if (mape < MAPE):

            MAPE = mape

            model2 = model

    plt.show()

    return mae, mape, model2

X, y = get\_dataset()

X, minsX, maxsX = normalize\_data(X)

y, minsy, maxsy = normalize\_data(y)

mae, mape, model = evaluate\_model(X, y)

model.save('ML.keras')

**Программа normalize.py**

def normalize\_data(X):

  nX=X.copy()

  minsX=[]

  maxsX=[]

  for j in range(0,X.shape[1]):

    minsX.append(min(X[:,j]))

    maxsX.append(max(X[:,j]))

    for i in range(0,X.shape[0]):

      nX[i,j]=(X[i,j]-minsX[j])/(maxsX[j]-minsX[j])\*0.9+0.1

  return nX,minsX,maxsX

def normalize\_one\_row(X, all\_data\_X):

  nX=X.copy()

  minsX=[]

  maxsX=[]

  for j in range(0,X.shape[1]):

    minsX.append(min(all\_data\_X[:,j]))

    maxsX.append(max(all\_data\_X[:,j]))

    for i in range(0,X.shape[0]):

      nX[i,j]=(X[i,j]-minsX[j])/(maxsX[j]-minsX[j])\*0.9+0.1

  return nX,minsX,maxsX

def denormalize\_data(X,minsX,maxsX):

  dX=X.copy()

  for j in range(0,X.shape[1]):

    for i in range(0,X.shape[0]):

      dX[i,j]=((X[i,j]-0.1)/0.9)\*(maxsX[j]-minsX[j])+minsX[j]

  return dX

**Программа predict.py**

import keras

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from normalize import normalize\_data, denormalize\_data, normalize\_one\_row

from get\_dataset import get\_dataset

model = keras.models.load\_model('ML.keras')

X, y = get\_dataset()

prX = np.array([[293.0, 4.0]])

prX, minPrX, maxPrX = normalize\_one\_row(prX, X)

X, minsX, maxsX = normalize\_data(X)

y, minsy, maxsy = normalize\_data(y)

new\_y=model.predict(X)

new\_y\_pr = model.predict(prX)

dnX = denormalize\_data(X, minsX, maxsX)

prDnX = denormalize\_data(prX, minsX, maxsX)

dny = denormalize\_data(y, minsy, maxsy)

new\_y = denormalize\_data(new\_y, minsy, maxsy)

new\_y\_pr = denormalize\_data(new\_y\_pr, minsy, maxsy)

fig, ax = plt.subplots(1, 2)

temps = []

ydata = [[], []]

ax[0].set\_title("P = const = 4.14 бар")

for i in range(119, 136):

    temps.append([dnX[i][0]])

    ydata[0].append(dny[i][0])

    ydata[1].append(new\_y[i][0])

ax[0].plot(temps, ydata[0], "ro")

ax[0].plot(temps, ydata[1])

ax[0].plot(prDnX[0][0], new\_y\_pr[0], "bo")

ax[1].set\_title("P = const = 2.72 бар")

temps = []

ydata = [[], []]

for i in range(85, 102):

    temps.append([dnX[i][0]])

    ydata[0].append(dny[i][0])

    ydata[1].append(new\_y[i][0])

ax[1].plot(temps, ydata[0], "ro")

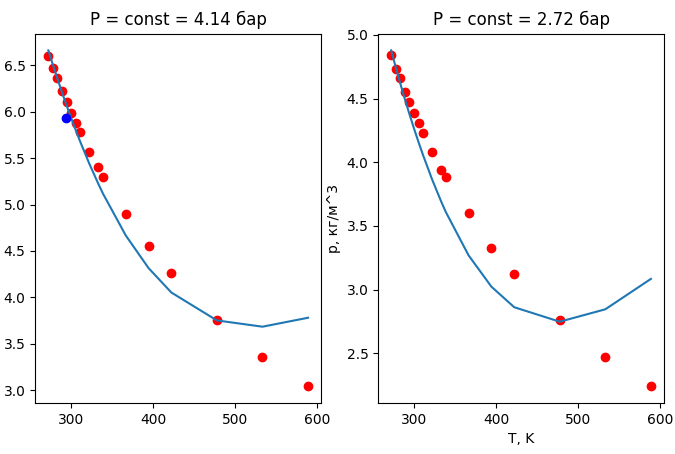
ax[1].plot(temps, ydata[1])

plt.ylabel("p, кг/м^3")

plt.xlabel("T, K")

plt.show()

**Результаты расчетов**



В результате работы программа *predict.py* выведет график зависимости плотности от температуры при постоянном давлении. Где синяя линия – предсказания модели, красные точки – экспериментальные данные, а синяя точка – точка, предсказанная нейросетью при T= 293 К, p = 4 бар.