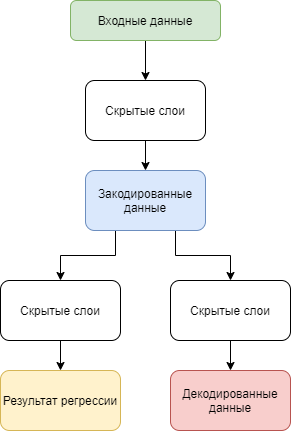
**Постановка задачи.**

Вариант 3\_6

Необходимо в зависимости от варианта сгенерировать датасет и сохранить его в формате csv.

Построить модель, которая будет содержать в себе автокодировщик и регрессионную модель. Схематично это должно выглядеть следующим образом:



Обучить модель и разбить обученную модель на 3: Модель кодирования данных (Входные данные -> Закодированные данные), модель декодирования данных (Закодированные данные -> Декодированные данные), и регрессионную модель (Входные данные -> Результат регрессии).

В качестве результата представить исходный код, сгенерированные данные в формате csv, кодированные и декодированные данные в формате csv, результат регрессии в формате csv (что должно быть и что выдает модель), и сами 3 модели в формате h5.

**Реализация**

В начале была написана функция для генерации датасета:

def generate(length):

data = np.zeros([length, 6])

x = np.random.normal(0, 10, length)

e = np.random.normal(0, 0.3, length)

data[:, 0] = np.array(x \*\* 2 + x + e)

data[:, 1] = np.array(np.fabs(x) + e)

data[:, 2] = np.array(np.sin(x - np.pi / 4) + e)

data[:, 3] = np.array(np.log(np.fabs(x)) + e)

data[:, 4] = np.array(-(x \*\* 3) + e)

data[:, 5] = np.array(-x + e)

labels = -x / 4 + e

return data, labels

Для обучения модели обучающий набор данных были сгенерирован размером 1000, а тестовый размером 100. Затем была составлена и обучена модель:

input = Input(shape=(6, ))

encoder = Dense(64, activation='relu')(\_input)

encoder = Dense(16, activation='relu')(encoder)

encoder = Dense(4, activation='relu')(encoder)

decoder = Dense(16, activation='relu')(encoder)

decoder = Dense(64, activation='relu')(decoder)

decoder = Dense(6)(decoder)

regress = Dense(16, activation='relu')(encoder)

regress = Dense(64, activation='relu')(regress)

regress = Dense(32, activation='relu')(regress)

regress = Dense(1)(regress)

model = Model(inputs=\_input, outputs=[decoder, regress])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

model.fit(train\_data, [train\_data, train\_labels], epochs=100, batch\_size=10)

Затем данная модель была разбита на 3 отдельных, через которые были пропущенные тестовые данные:

encoder\_m = Model(inputs=\_input, outputs=decoder)

decoder\_m = Model(inputs=\_input, outputs=encoder)

regress\_m = Model(inputs=\_input, outputs=regress)

encoder\_res = encoder\_m.predict(test\_data)

decoder\_res = decoder\_m.predict(test\_data)

regression\_res = regress\_m.predict(test\_data)

Полученные данные из переменных decoder\_res и regression\_res совпадают со входными данными из переменных test\_data и test\_labels соответственно.