

## Постановка задачи.

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).

Вариант 2, цель 3 (номер зачетки: 30)

$X \in N(-5,10)$   
 $e \in N(0,0.3)$

Признак	1	2	3	4	5	6	7
Формула	$-X^3+e$	$\ln( X )+e$	$\sin(3X)+e$	$\exp(X)+e$	$X+4+e$	$-X+\sqrt{ X }+e$	$X+e$

## Выполнение работы.

Генерация датасета происходит в модуле generate\_data.py. Создаются 1000 обучающих примеров (train.csv) и 200 тестовых (test.csv).

Задание модели:

```
main_input = Input(shape=(6,), name='main_input')
layer_encoder = Dense(36, activation='relu')(main_input)
layer_encoder = Dense(36, activation='relu')(layer_encoder)
layer_encoder = Dense(encoded_size, activation='relu',
name='encoder_output')(layer_encoder)

layer_decoder = Dense(36, activation='relu',
name='decoder_layer_0')(layer_encoder)
layer_decoder = Dense(36, activation='relu',
name='decoder_layer_1')(layer_decoder)
layer_decoder = Dense(6, name='decoder_output')(layer_decoder)

layer_regression = Dense(18, activation='relu')(layer_encoder)
layer_regression = Dense(36, activation='relu')(layer_regression)
layer_regression = Dense(36, activation='relu')(layer_regression)
layer_regression = Dense(18, activation='relu')(layer_regression)
layer_regression = Dense(1, name='regression_output')(layer_regression)
```

```
model = Model(inputs=[main_input], outputs=[layer_regression,
layer_decoder])
```

Три отдельные модели кодирования, декодирования и регрессии:

```
encoder_model = Model(main_input, layer_encoder)
regression_model = Model(main_input, layer_regression)
decoder_input = Input(shape=(4,))
dec = model.get_layer('decoder_layer_0')(decoder_input)
dec = model.get_layer('decoder_layer_1')(dec)
dec = model.get_layer('decoder_output')(dec)
decoder_model = Model(decoder_input, dec)
```

Сохранение моделей и результатов прогона:

```
encoder_model.save('encoder_model.h5')
decoder_model.save('decoder_model.h5')
regression_model.save('regression_model.h5')

encoder_predict = encoder_model.predict(test_data)
np.savetxt('encoded.csv', encoder_predict, delimiter=';')
np.savetxt('regression.csv', np.hstack((test_labels,
regression_model.predict(test_data))), delimiter=';')
np.savetxt('decoded.csv', decoder_model.predict(encoder_predict),
delimiter=';')
```

MAE на тестовых данных для регрессии 0.0511, для декодирования 0.3152.