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')

encoder = Dense(24, activation='relu')(main_input)
encoder = Dense(24, activation='relu')(encoder)
encoder = Dense(4, name='encoder_output')(encoder)

decoder = Dense(24, activation='relu', name='decoder_layer_0')(encoder)
decoder = Dense(24, activation='relu', name='decoder_layer_1')(decoder)
decoder = Dense(6, name='decoder_output')(decoder)

regression = Dense(18, activation='relu')(encoder)
regression = Dense(36, activation='relu')(regression)
regression = Dense(36, activation='relu')(regression)
regression = Dense(18, activation='relu')(regression)
regression = Dense(1, name='regression_output')(regression)


model = Model(inputs=[main_input], outputs=[regression, decoder])
```

```
model.compile(optimizer='adam', loss='mse', metrics='mae')
model.fit(train_data, [train_labels, train_data], epochs=100,
batch_size=50)
```

Три отдельные модели кодирования, декодирования и регресии:

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

encoder_predict = encoder_model.predict(test_data)
regression_predict = np.hstack((test_labels,
regression_model.predict(test_data)))
decoder_predict = decoder_model.predict(encoder_predict) * std + mean
np.savetxt('encoded.csv', encoder_predict, delimiter=';')
np.savetxt('regression.csv', regression_predict, delimiter=';')
np.savetxt('decoded.csv', decoder_predict, delimiter=';')
```