Был выполнен 1 вариант. В качестве цели регрессии использовалась функция 7 : $\frac{x^3}{4} + e$

Генерация обучающих и тестовых данных и запись их в csv файле:

```
def gen_data(size=1000, x_down=3, x_up=10, e_up=0.3):
    size1 = size // 2
    size2 = size - size1

    x1 = np.asarray([np.random.rand(1) * (x_up - x_down) + x_down for i in
range(size1)])
    y1 = np.asarray([(x ** 3)/4 + np.random.rand(1) * e_up for x in x1])

    x2 = np.asarray([np.random.rand(1) * (x_up - x_down) + x_down for i in
range(size2)])
    y2 = np.asarray([(x ** 3) / 4 + np.random.rand(1) * e_up for x in x2])

    return np.hstack((x1, y1)), np.hstack((x2, y2))


train, test = gen_data()
np.savetxt("train.csv", train, delimiter=";")
np.savetxt("test.csv", test, delimiter=";")
```

Получение данных из csv:

```
train_data = np.genfromtxt('train.csv', delimiter=';')
test_data = np.genfromtxt('test.csv', delimiter=';')

train_x = np.reshape(train_data[:, 0], (len(train_data), 1))
train_y = np.reshape(train_data[:, 1], (len(train_data), 1))
test_x = np.reshape(test_data[:, 0], (len(test_data), 1))
test_y = np.reshape(test_data[:, 1], (len(test_data), 1))
```

Допустим, в качестве кодирования используется следующее преобразование:

```
coded_train_x = train_x * 2
coded_test_x = test_x * 2
```

Архитектура модели:

```
main_input = Input(shape=(1,), name='main_input')

encoding_layer = Dense(16, activation='relu')(main_input)
encoding_output = Dense(1, activation='relu',
name='encoding_output')(encoding_layer)

decoding_layer = Dense(16, activation='relu')(encoding_output)
decoding_output = Dense(1, activation='relu',
name='decoding_output')(decoding_layer)

regression_layer = Dense(64, activation='relu')(encoding_output)
regression_layer = Dense(64, activation='relu')(regression_layer)
regression_layer = Dense(64, activation='relu')(regression_layer)
regression_output = Dense(1, name='regression_output')(regression_layer)
```

Обучение модели полностью:

```
model = Model(inputs=[main_input], outputs=[regression_output,
encoding_output, decoding_output])
model.compile(optimizer='rmsprop', loss='mse', metrics='mae')
model.fit([train_x], [train_y, coded_train_x, train_x], epochs=150,
batch_size=5, validation_split=0)
```

Создание и тестирование 3-ех моделей:

```
test = np.array([[3], [4], [5], [6], [7], [8], [9], [10]])

regression_model = Model(inputs=[main_input], outputs=[regression_output])
print(regression_model.predict(test))
regression_prediction = regression_model.predict(test_x)

encoding_model = Model(inputs=[main_input], outputs=[encoding_output])
print(encoding_model.predict(test))
encoding_prediction = encoding_model.predict(test_x)

decoding_model = Model(inputs=[main_input], outputs=[decoding_output])
print(decoding_model.predict(test))
decoding_prediction = decoding_model.predict(test_x)
```

Результаты приведены в csv файлах.

Сохранение моделей и результатов:

```
regression_model.save('regression_model.h5')
encoding_model.save('encoding_model.h5')
decoding_model.save('decoding_model.h5')

np.savetxt('regression_results.csv', np.hstack((test_y,
regression_prediction)), delimiter=';')
np.savetxt('encoding_results.csv', np.hstack((coded_test_x,
encoding_prediction)), delimiter=';')
np.savetxt('decoding_results.csv', np.hstack((test_x, decoding_prediction)),
delimiter=';')
```