Вариант 1 (29 % 7 + 1 = 2)

Формула: $sin\left(\frac{x}{2}\right) + e, x \in N(3, 10),\ e \in N(0, 0.3)$

Выполнение работы.

Был написана программа, генерирующая тренировочный и тестовый датасет. Код программы находится в файле generateDataset.py.

```
x = np.random.normal(3.0, 10.0, 3000).reshape(-1,1) # генерируем x
e = np.random.normal(0, 0.3, 3000).reshape(-1,1)  # генерируем ошибку

y = np.sin(x/2) + e # считаем y

np.savetxt('train_dataset.csv', np.hstack((x, y))) # записываем в файл

x = np.random.normal(3.0, 10.0, 1000).reshape(-1,1)
e = np.random.normal(0, 0.3, 1000).reshape(-1,1)

y = np.sin(x/2) + e

np.savetxt('test_dataset.csv', np.hstack((x, y)))
```

Была создана модель, согласно схеме.

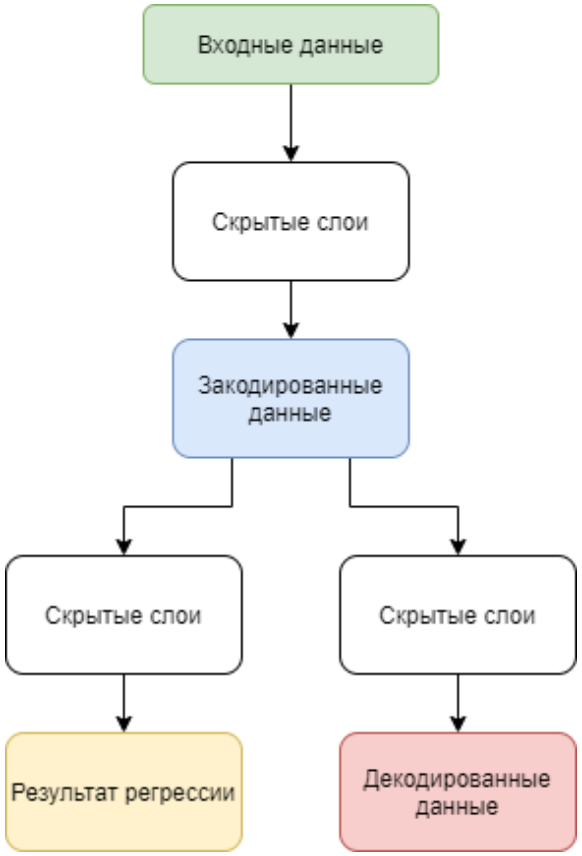

Рисунок 1 – Схема модели

Модель:

```
inp = Input(shape=(1,))
```

```
# encoder
dense_ecoder = Dense(1, activation='relu', name='dense_ecoder')(inp)

# regression
dense_1 = Dense(64, activation='relu')(dense_ecoder)
dense_2 = Dense(64, activation='relu')(dense_1)
out_reg = Dense(1, name='out_reg')(dense_2)

# decoder
dense_decoder = Dense(1, name='dense_decoder')(dense_ecoder)

# configuration
model = Model(inputs=inp, outputs=[out_reg, dense_decoder, dense_ecoder])
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
model.fit(train_data, [train_targets, train_data, train_data *
CONST_FOR_ENCODE], epochs=200, batch_size=10)
```

Кодировать данные будем путем умножения входных данных на константу, равную 5. Разобьем модель на 3 и прогоним через каждую тестовые данные. Результаты для каждой модели запишем в файлы. Так же сохраним каждую из 3х моделей.

```
# encoder model
ecoder_model = Model(inputs=inp, outputs=dense_ecoder)
encoder_pred = np.asarray(ecoder_model.predict(test_data))
np.savetxt('outOfModels/data_encoder_model.csv', np.hstack((test_data *
CONST_FOR_ENCODE, encoder_pred)))
ecoder_model.save('models/encoder_model.h5')

# regression model
reg_model = Model(inputs=inp, outputs=out_reg)
reg_pred = np.asarray(reg_model.predict(test_data))
np.savetxt('outOfModels/data_reg_model.csv', np.hstack((test_targets,
reg_pred)))
ecoder_model.save('models/reg_model.h5')

# decoder model
decoder_model = Model(inputs=inp, outputs=dense_decoder)
decoder_pred = np.asarray(reg_model.predict(test_data * CONST_FOR_ENCODE))
np.savetxt('outOfModels/data_decoder_model.csv', np.hstack((test_data,
decoder_pred)))
ecoder_model.save('models/decoder_model.h5')
```

Результаты запусков показали, что модели работаю плохо. В подавляющем большенстве случаев предсказания модели имеют большую погрешность. Предполагаю, что это связано с обучением модели, так как 3 части большой модели мешают друг другу обучаться. При обучении частей по отдельности, каждая часть показывает хорошие результаты.