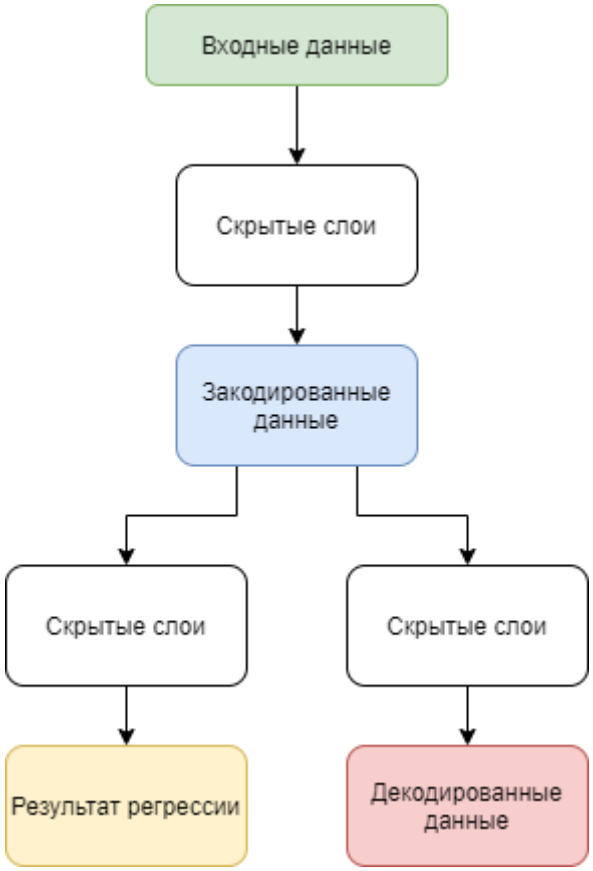
Постановка задачи.

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.