Кобенко В.П. 8382 Практика 5. Вариант 4. Цель регрессии X^2+e (номер зачетки оканчивается на 04)

Задание:

Вариант 4

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

Признак	1	2	3	4	5	6	7
Формула	cos(X)+e	-X+e	sin(X)*X+e	sqrt(|X|)+e	X^2+e	-|X|+4	X-(X^2)/5+e

Выполнение работы:

Создание датасета (и его параметров):

```
# Параметры данных
DIM_DATASET = 6
SIZE_TRAINING = 400
SIZE_TEST = 100

# Генерирование данных
def generation_of_dataset(size_of_dataset):
    dataset = np.zeros((size_of_dataset, DIM_DATASET))
    dataset_y = np.zeros(size_of_dataset)
    for i in range(size_of_dataset):
        X = np.random.normal(0, 10)
        e = np.random.normal(0, 0.3)
        dataset[i, :] = (np.round(np.cos(X) + e), np.round(-X + e), np.round(np.sin(X) * X + e), np.round(np.sqrt(np.fabs(X)) + e), np.round(-np.fab
        dataset_y[i] = np.round(X ** 2 + e)
    return np.round(np.array(dataset), decimals=3), np.array(dataset_y)
```

Создание Encoder:

```
# Encoder
main_input = Input(shape=(DIM_DATASET,), name='main_input')
encoded = Dense(64, activation='relu')(main_input)
encoded = Dense(32, activation='relu')(encoded)
encoded = Dense(DIM_DATASET, activation='linear')(encoded)
```

Создание Decoder:

```
# Decoder
input_encoded = Input(shape=(DIM_DATASET,), name='input_encoded')
decoded = Dense(32, activation='relu', kernel_initializer='normal')(input_encoded)
decoded = Dense(64, activation='relu')(decoded)
decoded = Dense(DIM_DATASET, name="out_aux")(decoded)
```

Создание Регрессии(результат):

```
# Regression
predicted = Dense(64, activation='relu', kernel_initializer='normal')(encoded)
predicted = Dense(32, activation='relu')(predicted)
predicted = Dense(16, activation='relu')(predicted)
predicted = Dense(1, name="out_main")(predicted)
```

Модели:

```
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    return model
```

```
encoded = Model(main_input, encoded, name="encoder")
decoded = Model(input_encoded, decoded, name="decoder")
predicted = Model(main_input, predicted, name="regr")

return encoded, decoded, predicted, main_input
```

Обучение сети:

```
encoded, decoded, full_model, main_input = create_objects()

# Регрессионная модель
full_model.compile(optimizer="adam", loss="mse", metrics=['mae'])

# Обучение сети
History = full_model.fit(x_train, y_train, epochs=40, batch_size=5, verbose=1, validation_data=(x_test, y_test))

encoded_data = encoded.predict(x_test)
decoded_data = decoded.predict(encoded_data)

regr = full_model.predict(x_test)
```

Запись результата:

```
# Запись в CSV необходимых данных
pd.DataFrame(np.round(regr, 3)).to_csv("result.csv")
pd.DataFrame(np.round(x_test, 3)).to_csv("x_test.csv")
pd.DataFrame(np.round(y_test, 3)).to_csv("y_test.csv")
pd.DataFrame(np.round(x_train, 3)).to_csv("x_train.csv")
pd.DataFrame(np.round(y_train, 3)).to_csv("y_train.csv")
pd.DataFrame(np.round(encoded_data, 3)).to_csv("encoded.csv")
pd.DataFrame(np.round(decoded_data, 3)).to_csv("decoded.csv")

# Сохранение модели
decoded.save('decoder.h5')
encoded.save('encoder.h5')
full_model.save('full.h5')
```