Шишкин Иван, гр. 8383

Вариант 4 признак 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.

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.

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.

Был сгенерирован датасет (файл dataset_generation.py): 1000 тренировочных и 500 тестовых. Тренировочные сохранены в файл my_dataset_train.csv, тестовые – my_dataset_test.csv.

Затем, датасет в main.py был загружен из этих файлов. Первые 6 столбцов – данные, седьмой столбец – лэйблы. Модель представлена в листинге 1.

Листинг 1 – Создание модели

```
n_inputs = 6
```

```
# define encoder
visible = Input(shape=(n_inputs,))
e_l1 = Dense(32, activation='relu')(visible)
e_l2 = Dense(16, activation='relu')(e_l1)
e = Dense(4, name='encoder')(e_l2)
# define decoder
d_l1 = Dense(16, activation='relu', name='d_l1')(e)
d_l2 = Dense(32, activation='relu', name='d_l2')(d_l1)
d = Dense(n_inputs, name='decoder')(d_l2)
# output layer
l1 = Dense(64, activation='relu')(e)
l2 = Dense(32, activation='relu')(l1)
output = Dense(1, activation='linear', name='output')(l2)
# define autoencoder model
model = Model(inputs=visible, outputs=[output, d])
# compile autoencoder model
model.compile(optimizer='adam', loss='mse', metrics='mae')
model.fit(train_X, [train_Y, train_X], epochs=250, batch_size=5)
```

Создание моделей для кодирования, декодирования (модель декодирования данных на вход получает закодированные данные, а на выход – декодированные) и регрессии представлено в листинге 2.

Листинг 2 – Модели кодирования, декодирования, регрессии

```
e_model = Model(inputs=visible, outputs=e)
d_input = Input(shape=(4,))  # encoder output = 4
d_model_l1 = model.get_layer('d_l1')(d_input)
d_model_l2 = model.get_layer('d_l2')(d_model_l1)
d_model_output = model.get_layer('decoder')(d_model_l2)
d_model = Model(inputs=d_input, outputs=d_model_output)
reg_model = Model(inputs=visible, outputs=output)
```

Результаты предсказаний моделей на тестовых данных сохранены в файлы, как и сами модели (листинг 3).

Листинг 3 – Сохранение моделей

```
np.savetxt('encoded_model_prediction',          e_model.predict(test_X),
delimiter='\t')
np.savetxt('decoded_model_prediction',
d_model.predict(e_model.predict(test_X)), delimiter='\t')
```

```
np.savetxt('regression_model_prediction',            np.hstack((test_Y,
reg_model.predict(test_X))), delimiter='\t')

e_model.save('encoded_model.h5')
d_model.save('decoded_model.h5')
reg_model.save('regression_model.h5')
```