Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «Языковые процессы интеллектуальных систем»

Тема: «**Предобучение нейронных сетей с использованием автоэнкодерного подхода**»

Выполнил:

Студент 4 курса

Группы ИИ-23

Романюк А. П.

Проверила:

Андренко К.В.

Брест 2025

Цель: научиться осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода

Общее задание

1. Взять за основу любую сверточную или полносвязную архитектуру с количеством слоев более 3. Осуществить ее обучение (без предобучения) в соответствии с вариантом задания. Получить оценку эффективности модели, используя метрики, специфичные для решаемой задачи (например, MAPE – для регрессионной задачи или F1/Confusion matrix для классификационной).
2. Выполнить обучение с предобучением, используя автоэнкодерный подход, алгоритм которого изложен в лекции. Условие останова (например, по количеству эпох) при обучении отдельных слоев с использованием автоэнкодера выбрать самостоятельно.
3. Сравнить результаты, полученные при обучении с/без предобучения, сделать выводы.
4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

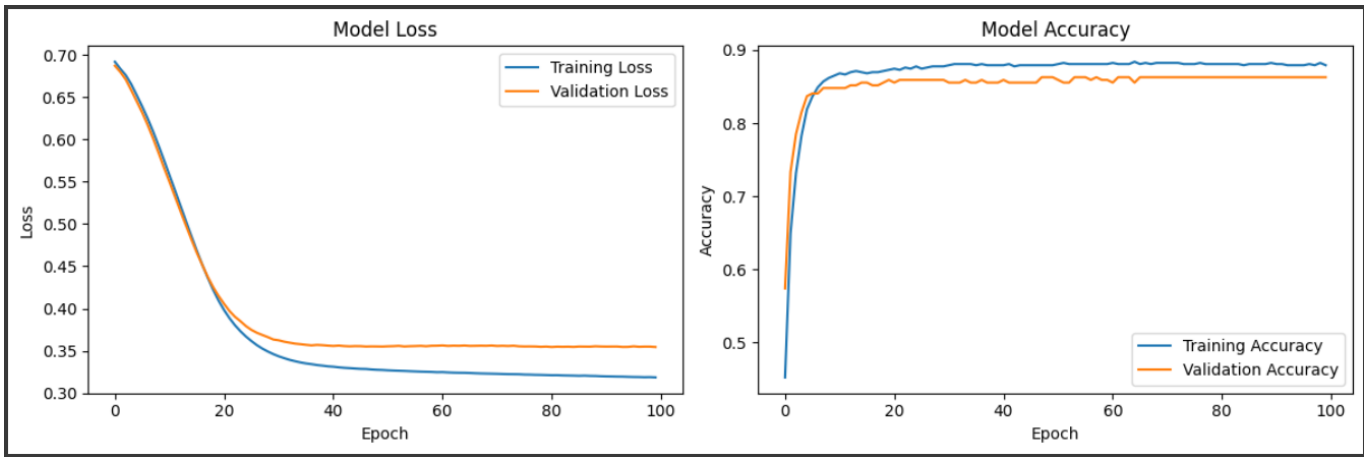
Задание по вариантам

9	https://archive.ics.uci.edu/dataset/850/raisin	классификация	Class

Код программы:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import matplotlib.pyplot as plt

data = pd.read_excel("Raisin_Dataset.xlsx")

label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

X = data.drop('Class', axis=1)
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential([
    Dense(64, input_shape=(X_train.shape[1],), activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')
```

```
])

model.compile(optimizer=Adam(learning_rate=0.00005), loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test), batch_size=4)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```



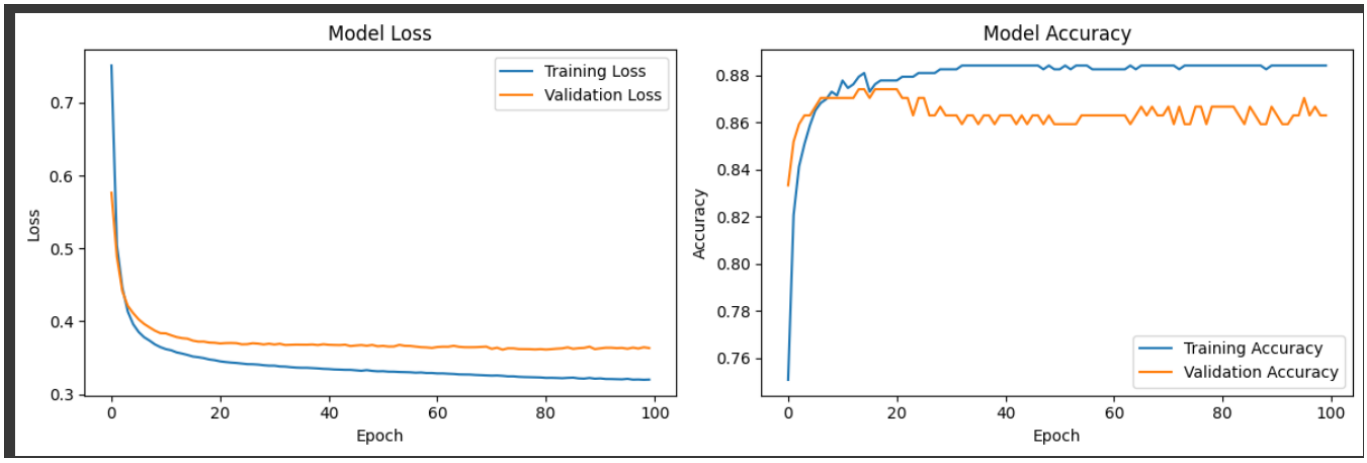

```
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

y_pred = np.argmax(model.predict(X_test), axis=-1)
```

```
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       129
           1       0.88      0.86      0.87       141

    accuracy                           0.86       270
   macro avg       0.86      0.86      0.86       270
weighted avg       0.86      0.86      0.86       270

Confusion Matrix:
 [[112  17]
 [ 20 121]]
```

```
from keras.layers import Input
from keras.models import Model

def build_autoencoder(input_dim, encoding_dim):
    input_layer = Input(shape=(input_dim,))
    encoded = Dense(encoding_dim, activation='relu')(input_layer)
    decoded = Dense(input_dim, activation='sigmoid')(encoded)
    autoencoder = Model(input_layer, decoded)
    encoder = Model(input_layer, encoded)
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder, encoder

autoencoder1, encoder1 = build_autoencoder(X_train.shape[1], 64)
autoencoder1.fit(X_train, X_train, epochs=100, batch_size=4, shuffle=True)

X_train_encoded = encoder1.predict(X_train)
autoencoder2, encoder2 = build_autoencoder(64, 32)
autoencoder2.fit(X_train_encoded, X_train_encoded, epochs=100, batch_size=4, shuffle=True)

X_train_encoded = encoder2.predict(X_train_encoded)
autoencoder3, encoder3 = build_autoencoder(32, 16)
autoencoder3.fit(X_train_encoded, X_train_encoded, epochs=100, batch_size=4, shuffle=True)

model.layers[0].set_weights(autoencoder1.get_weights()[:2])
model.layers[1].set_weights(autoencoder2.get_weights()[:2])
model.layers[2].set_weights(autoencoder3.get_weights()[:2])

history = model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test), batch_size=4)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
```

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

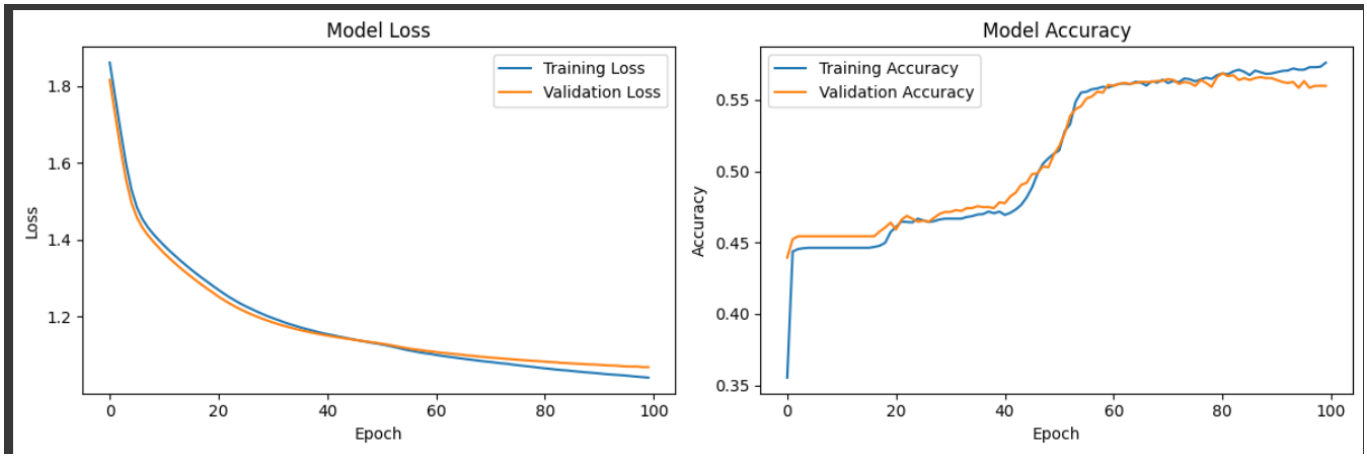



```
y_pred_pretrained = np.argmax(model.predict(X_test), axis=-1)

print("Classification Report (Pretrained):\n", classification_report(y_test, y_pred_pretrained))
print("Confusion Matrix (Pretrained):\n", confusion_matrix(y_test, y_pred_pretrained))
```

```
Classification Report (Pretrained):
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       129
           1       0.88      0.86      0.87       141

    accuracy                           0.86       270
   macro avg       0.86      0.86      0.86       270
weighted avg       0.86      0.86      0.86       270

Confusion Matrix (Pretrained):
 [[112  17]
 [ 20 121]]
```

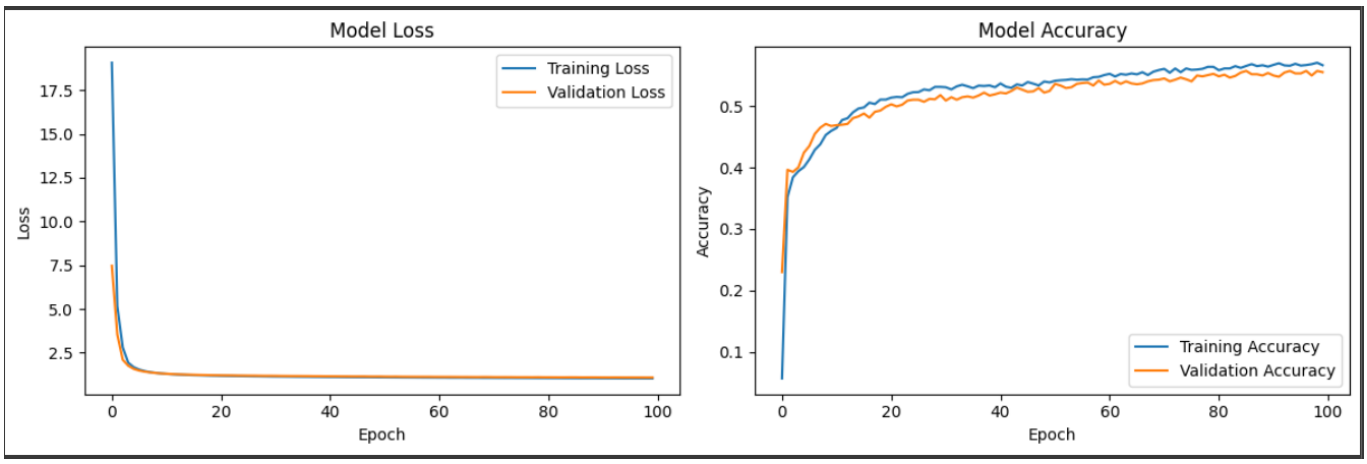
Данные для датасета winequality-white:

С нуля:




```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.00      0.00      0.00        40
           2       0.59      0.65      0.62       426
           3       0.55      0.69      0.61       668
           4       0.52      0.29      0.37       280
           5       0.00      0.00      0.00        49

    accuracy                           0.56      1470
   macro avg       0.28      0.27      0.27      1470
weighted avg       0.52      0.56      0.53      1470

Confusion Matrix:
 [[  0   0   2   5   0   0]
 [  0   0  24  16   0   0]
 [  0   0 279 142   5   0]
 [  0   0 153 464  51   0]
 [  0   0  10 190  80   0]
 [  0   0   1  29  19   0]]
```

Предтренированная:




```
Classification Report (Pretrained):
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         7
           1       0.00      0.00      0.00        40
           2       0.61      0.57      0.59       426
           3       0.54      0.74      0.63       668
           4       0.50      0.28      0.36       280
           5       0.00      0.00      0.00        49

    accuracy                           0.56      1470
   macro avg       0.28      0.27      0.26      1470
weighted avg       0.52      0.56      0.52      1470

Confusion Matrix (Pretrained):
 [[  0   0   2   5   0   0]
 [  0   0  23  17   0   0]
 [  0   3 241 175   7   0]
 [  0   0 119 497  52   0]
 [  0   0   7 194  79   0]
 [  0   0   1  29  19   0]]
```

Вывод: научился осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода