Ввод [5]:

```
class Relu_NN(nn.Module):
    def __init__(self):
        super().__init__()
        self.Matrix1 = nn.Linear(45,16)
        self.Matrix2 = nn.Linear(16,8)
        self.Matrix3 = nn.Linear(8,1)
        self.R = nn.ReLU() # changed to nn.Sigmoid() and nn.Softmax() after

    def forward(self,x):
        x = self.R(self.Matrix1(x))
        x = self.R(self.Matrix2(x))
        x = self.Matrix3(x)

        return x.squeeze()
```

Ввод [5]:

```
class new_nn1(nn.Module):
    def __init__(self):
        super().__init__()
        self.Matrix1 = nn.Linear(45,16)
        self.Matrix2 = nn.Linear(16,8)
        self.Matrix3 = nn.Linear(8,4)
        self.R1 = nn.ReLU()
        self.Matrix4 = nn.Linear(4,2)
        self.Matrix5 = nn.Linear(2,1)
        self.R2 = nn.ReLU()


    def forward(self,x):
        x = self.R1(self.Matrix1(x))
        x = self.R1(self.Matrix2(x))
        x = self.R1(self.Matrix3(x))
        x = self.R2(self.Matrix4(x))
        x = self.Matrix5(x)

        return x.squeeze()
```

Ввод [5]:

```
class new_nn2(nn.Module):
    def __init__(self):
        super().__init__()
        self.Matrix1 = nn.Linear(45,16)
        self.Matrix2 = nn.Linear(16,8)
        self.Matrix3 = nn.Linear(8,4)
        self.R1 = nn.LeakyReLU()
        self.Matrix4 = nn.Linear(4,2)
        self.Matrix5 = nn.Linear(2,1)
        self.R2 = nn.LeakyReLU()


    def forward(self,x):
        x = self.R1(self.Matrix1(x))
        x = self.R1(self.Matrix2(x))
        x = self.R1(self.Matrix3(x))
        x = self.R2(self.Matrix4(x))
        x = self.Matrix5(x)

        return x.squeeze()
```

Ввод [3]:

```
from sklearn.model_selection import train_test_split

class MyDataset(Dataset):
    def __init__(self, filepath, selected_headers, test_size=0.2, random_state=42):
        self.data = pd.read_csv(filepath, header=0)
        self.data = self.data.drop(labels=["id"], axis=1)
        self.headers = self.data.columns.tolist()

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(
            self.data[selected_headers].values,
            self.data[self.headers[-1]].values,
            test_size=test_size,
            random_state=random_state
        )

        self.x_train = torch.tensor(X_train)
        self.y_train = torch.tensor(y_train)
        self.x_test = torch.tensor(X_test)
        self.y_test = torch.tensor(y_test)

    def __len__(self):
        return len(self.x_train)

    def __getitem__(self, idx):
        return self.x_train[idx].double(), self.y_train[idx].double()
```