Wydział Informatyki i Telekomunikacji

Sztuczna Inteligencja i Inżynieria Wiedzy

Lista nr 5

Jakub Krupiński
255356

1. Wprowadzenie

W wykonywanym ćwiczeniu przeprowadzam analizę zestawu danych Jester dotyczącego oceny śmieszności żartów. Wykorzystałem tu zestaw nr 1, który posiadał w sumie z 3 plików excel zawierających oceny żartów każdego z użytkowników oraz 100 plików HTML zawierających treści żartów, gdzie numery w nazwie pliku odnosiły się do indeksów żartów w plikach excel. Analiza została przeprowadzona za pomocą biblioteki *scikit-learn*, za pomocą sieci neuronowej MLP. Zadanie zostało wykonane za pomocą dziennika Jupyter.

2. Testowanie modeli

Modele testowane są dla następujących scenariuszy:

- Domyślna konfiguracja hiperparametrów:

```
def run(mode, _learning_rate='constant', _learning_rates_init=0.001, _hidden_layer_sizes=(100,), _random_state=None):
    mlp = MLPRegressor(
        solver='sgd',
        alpha=0.0,
        learning_rate=_learning_rate,    # constant, invscaling, adaptive',
        learning_rate_init=_learning_rates_init,   # default step (how much weight sizes are adjusted each time)
        hidden_layer_sizes=_hidden_layer_sizes,    # number of neurons in each layer
        random_state=_random_state,
    )
```

- Badanie wpływu tempa uczenia na otrzymane wyniki:

```
learning_rates_init_values = [0.0001, 0.001, 0.01]
learning_rates_values = ['constant', 'invscaling', 'adaptive']
```

- Badanie wpływu rozmiaru modelu MLP na otrzymywane wyniki:

```
hidden_layer_sizes_values = [(10,), (50,), (100,), (500,),
                             (10, 10), (50, 50,), (100, 100,), (500, 500,)]
```

- Badanie wpływu regularyzacji na otrzymywane wyniki:

```
alpha_values=[0.0, 0.0001, 0.001, 0.01, 0.1, 1.0, 2.5, 5.0]
```

- Testowanie najlepszej konfiguracji modelu MLP na własnych dowcipach:

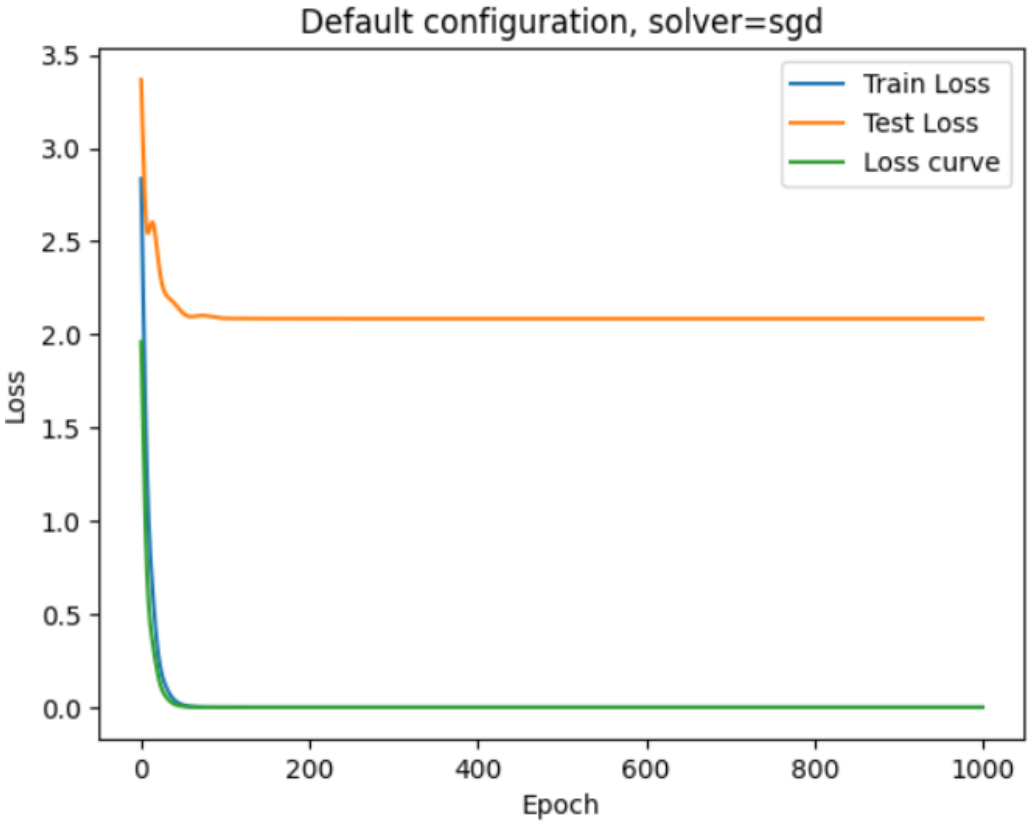
Za pomocą metody ‘run’:

```python
def run(mode, _learning_rate='constant', _learning_rates_init=0.001, _hidden_layer_sizes=(100,), _random_state=None, _alpha=0.0):
    mlp = MLPRegressor(
        solver='sgd',
        alpha=_alpha,
        learning_rate=_learning_rate,    # constant, invscaling, adaptive',
        learning_rate_init=_learning_rates_init,   # default step (how much weight sizes are adjusted each time)
        hidden_layer_sizes=_hidden_layer_sizes,    # number of neurons in each layer
        random_state=_random_state,
    )

    train_loss = []
    test_loss = []
    epochs_values = 1000

    for _ in range(epochs_values):
        mlp.partial_fit(x_train_scaled, y_train)
        pred_y_train = mlp.predict(x_train_scaled)
        pred_y_test = mlp.predict(x_test_scaled)

        train_loss.append(mean_squared_error(y_train, pred_y_train))
        test_loss.append(mean_squared_error(y_test, pred_y_test))
    loss_curve = mlp.loss_curve_

    if mode == 'train':
        return train_loss
    elif mode == 'test':
        return test_loss
    else:
        return loss_curve
```

Rozwiązanie to nie należy do optymalnych pod względem czasowym, jako że zwraca train loss, test loss lub loss curve pojedynczo, co wymaga ruszenia modelu trzykrotnie za każdym razem, podjąłem jednak decyzję, żeby zaimplementować to w ten sposób, jako że ułatwiło mi to tworzenie odpowiednich dla mnie wykresów.

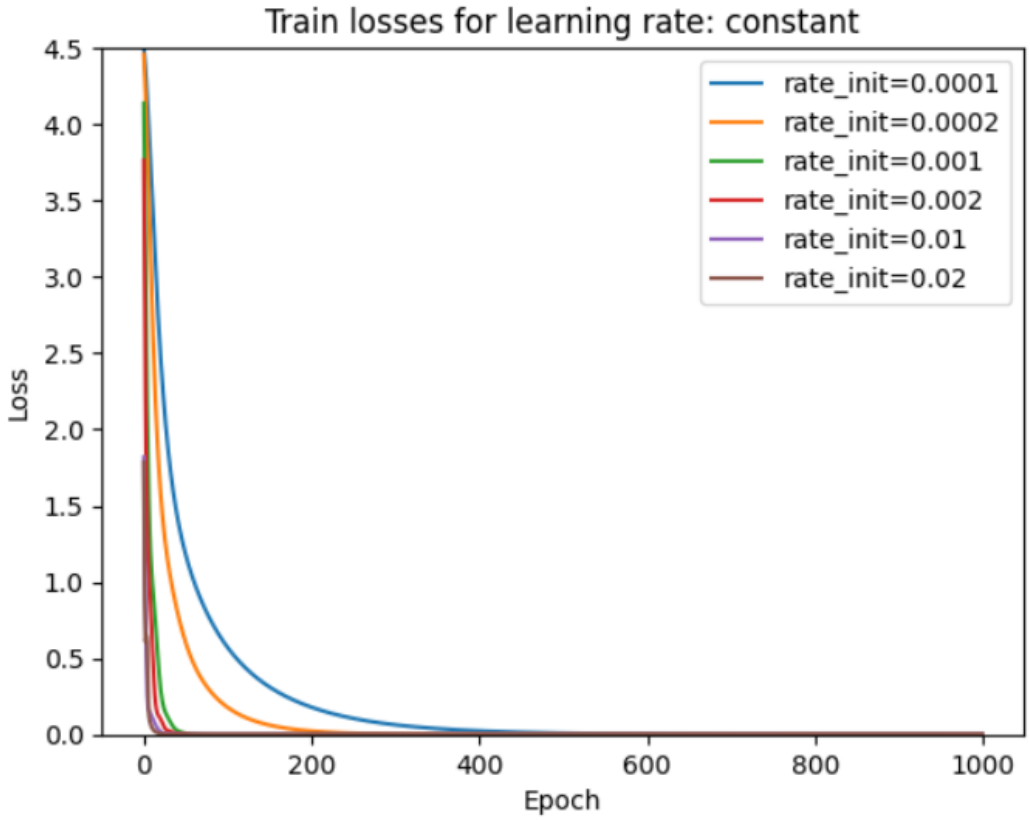
3. Analiza wyników

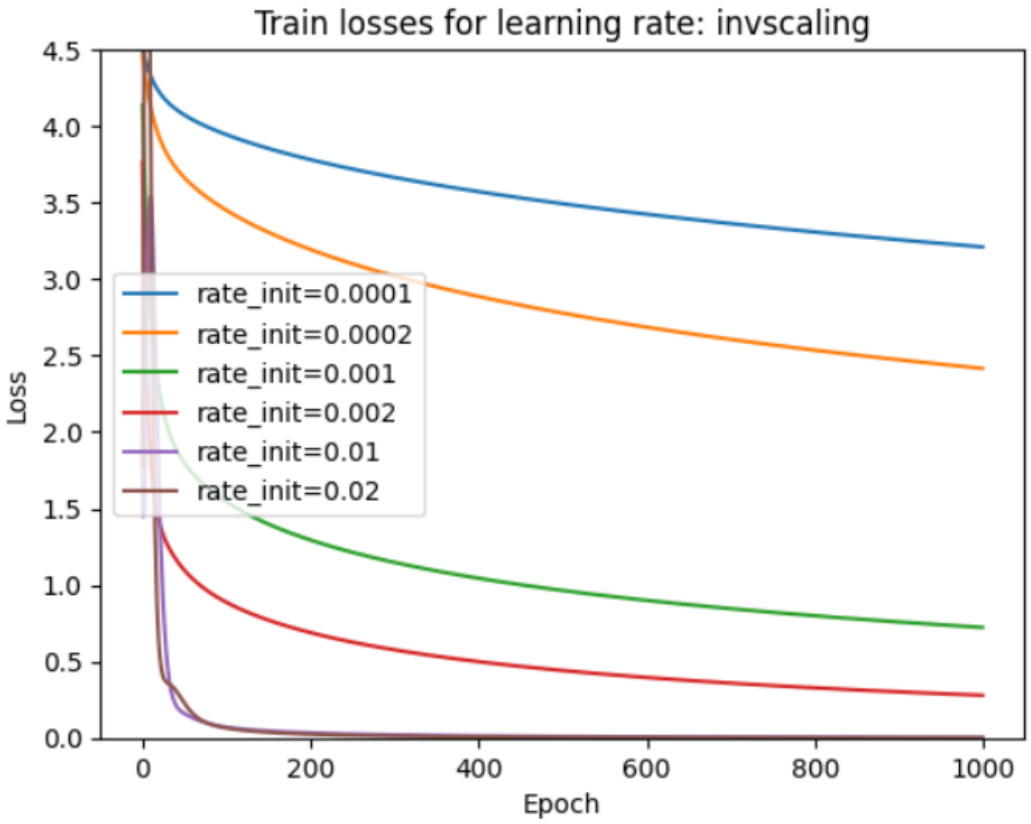
- Domyślna konfiguracja hiperparametrów

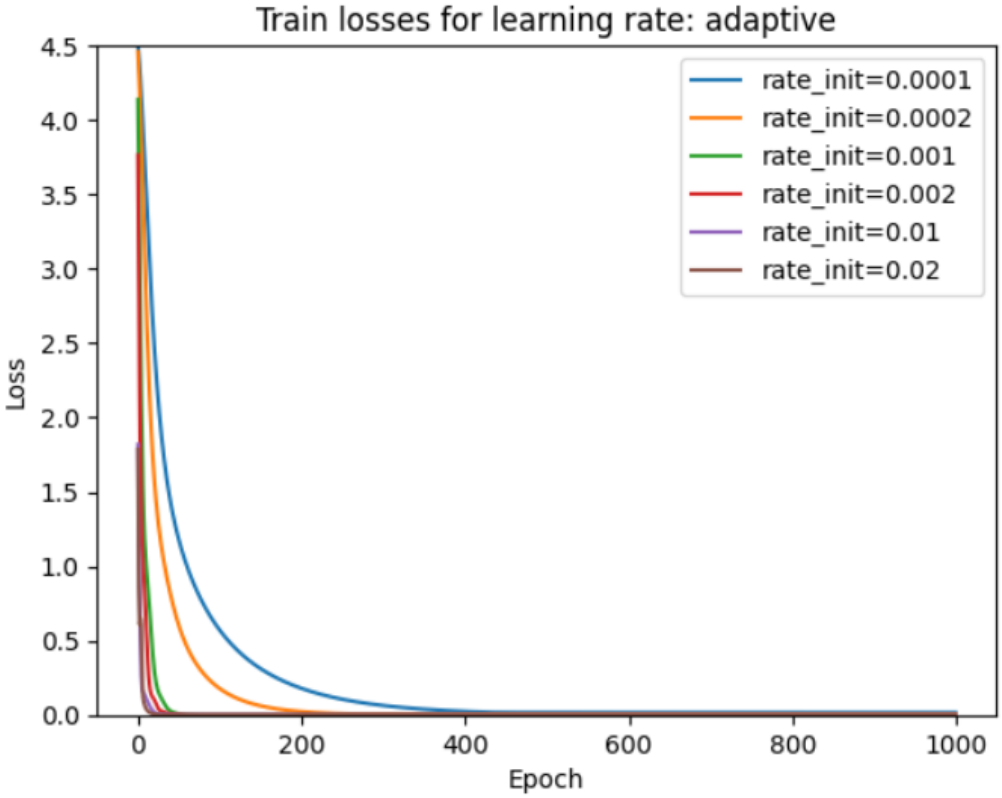


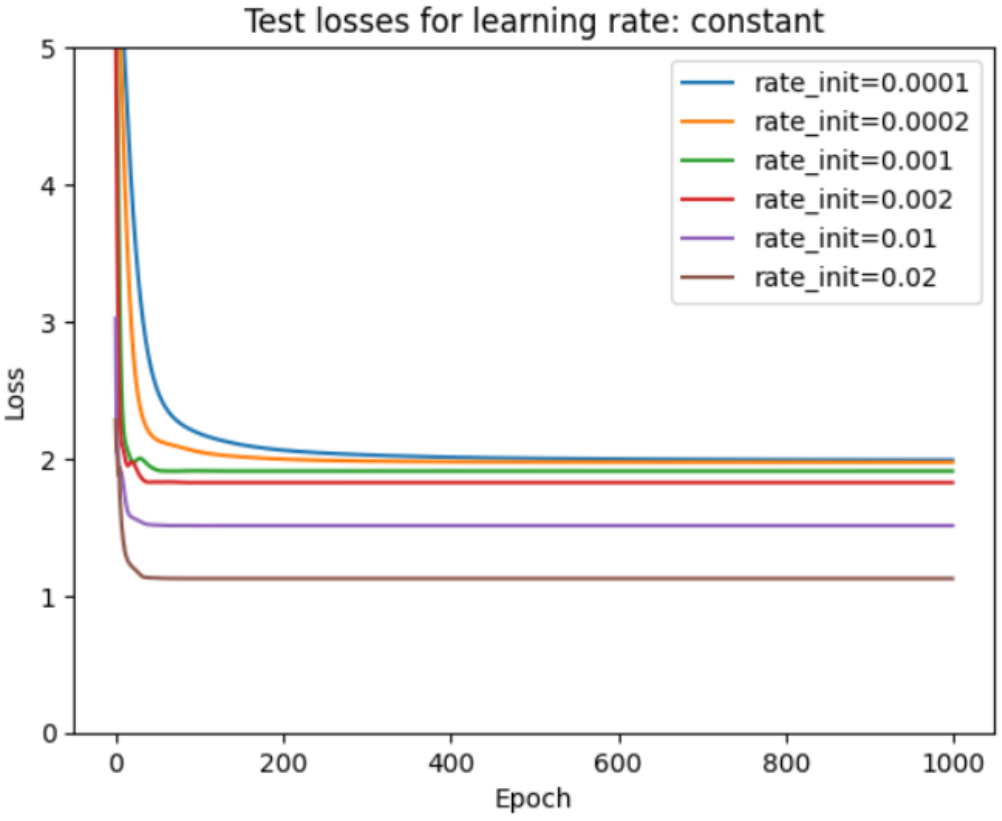
Model szybko osiągają wartości stałe, niezmieniające się w przyszłych epokach – model jest słabo dopasowany do danych treningowych i nie jest w stanie nauczyć się wzorców w danych treningowych

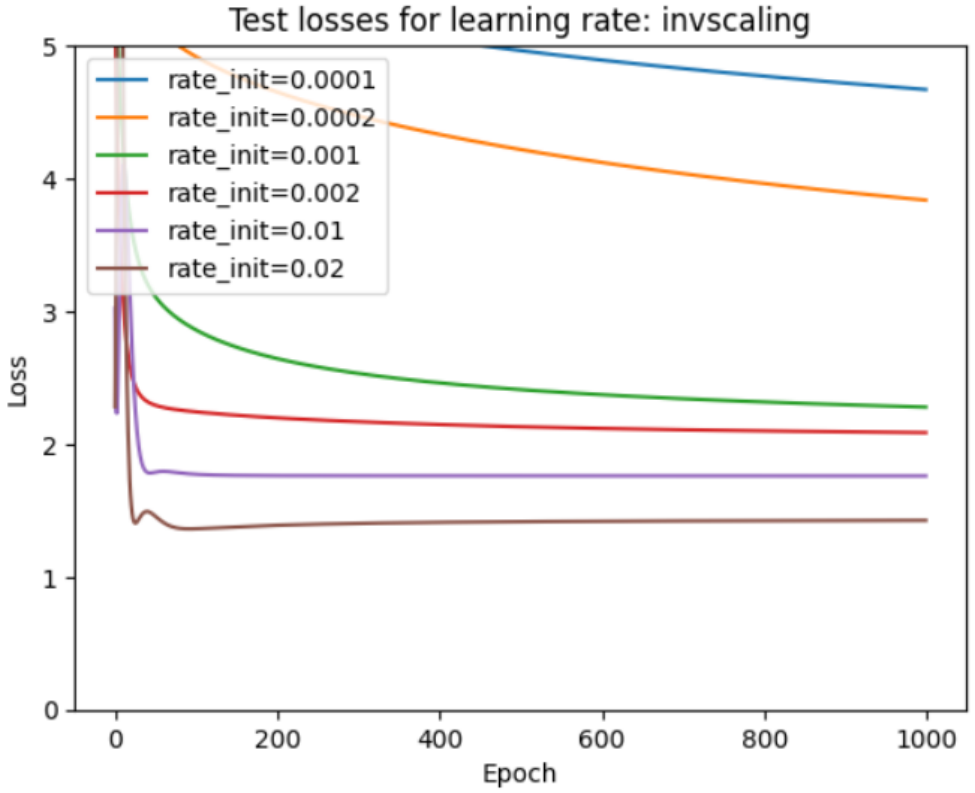
- Badanie wpływu tempa uczenia na otrzymane wyniki

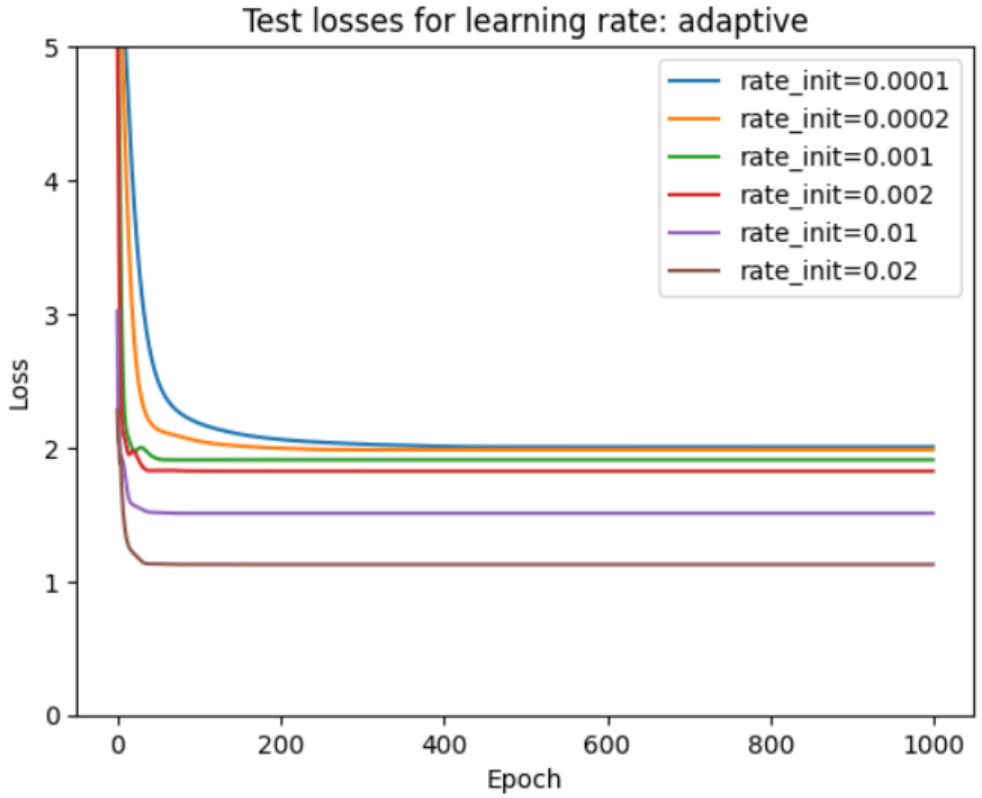


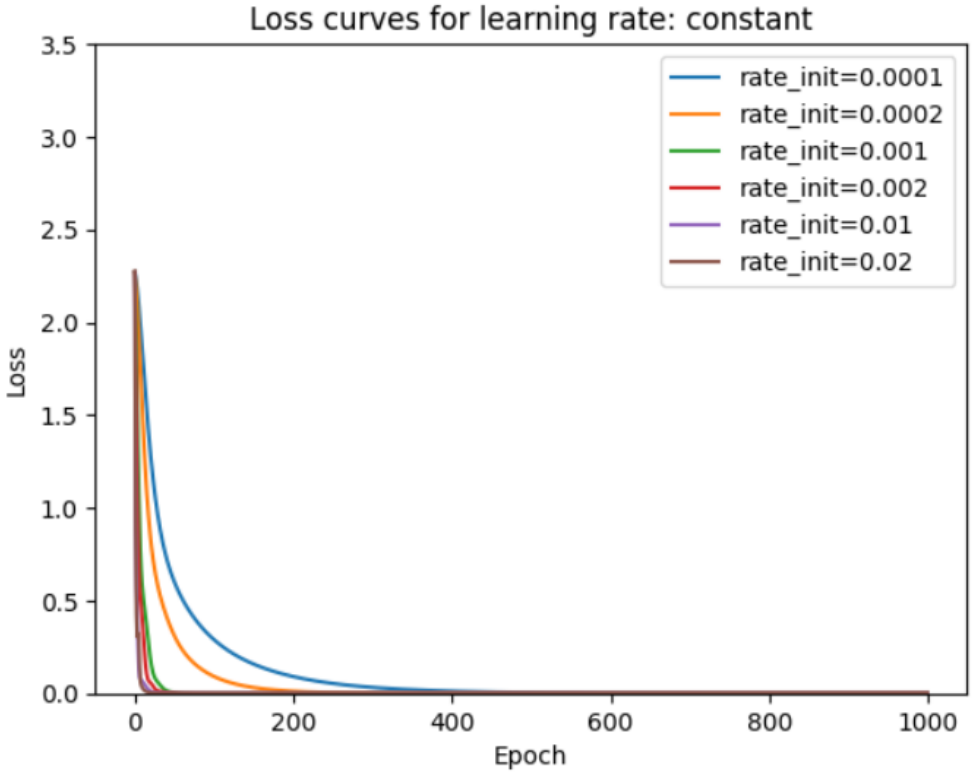


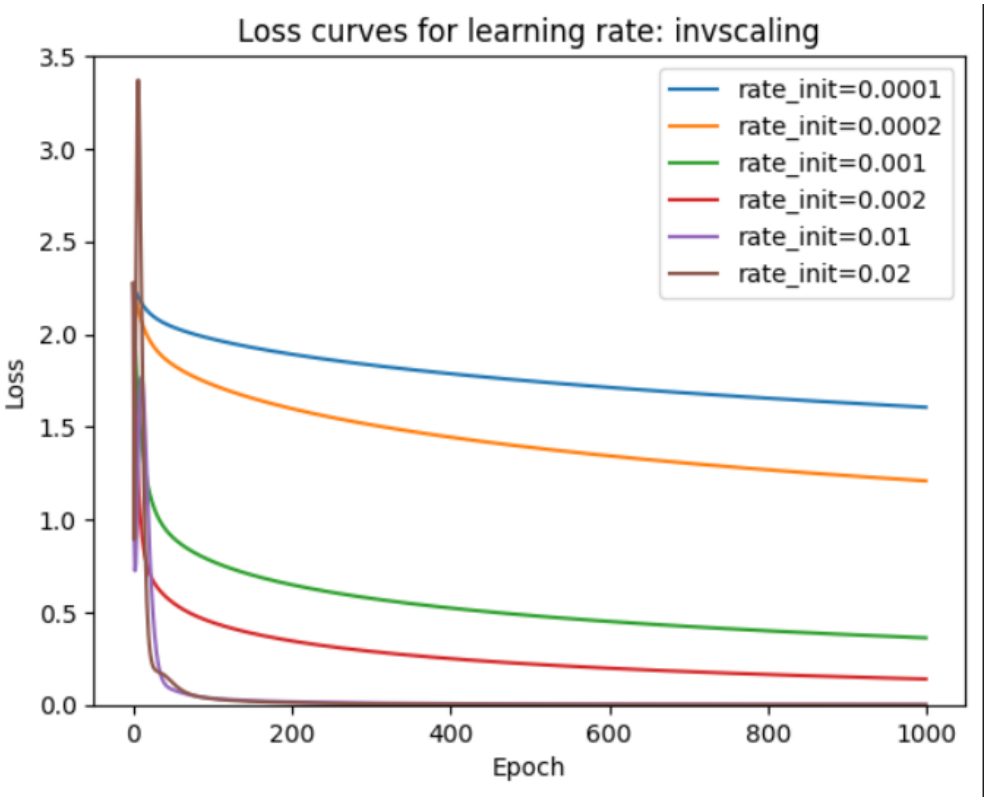

Train losses for learning rate: adaptive
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
0
200
400
600
800
1000

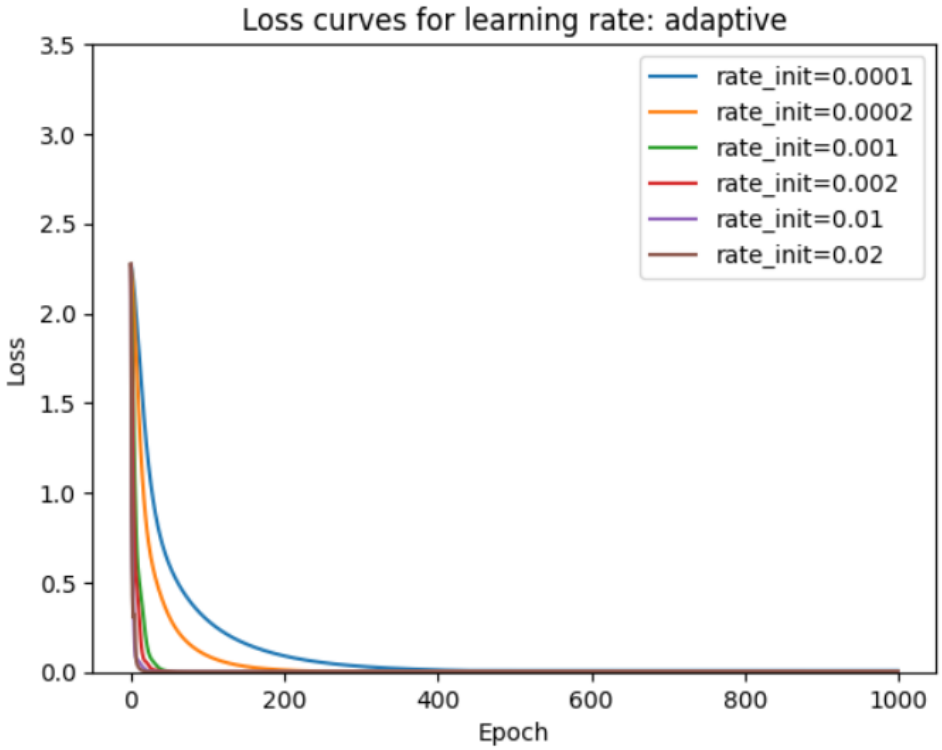

Test losses for learning rate: constant
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0
1
2
3
4
5
0
200
400
600
800
1000


Test losses for learning rate: invscaling
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0
1
2
3
4
5
0
200
400
600
800
1000


Test losses for learning rate: adaptive
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0
1
2
3
4
5
0
200
400
600
800
1000


Loss curves for learning rate: constant
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000


Loss curves for learning rate: invscaling
rate_init=0.0001
rate_init=0.0002
rate_init=0.001
rate_init=0.002
rate_init=0.01
rate_init=0.02
Loss
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
200
400
600
800
1000

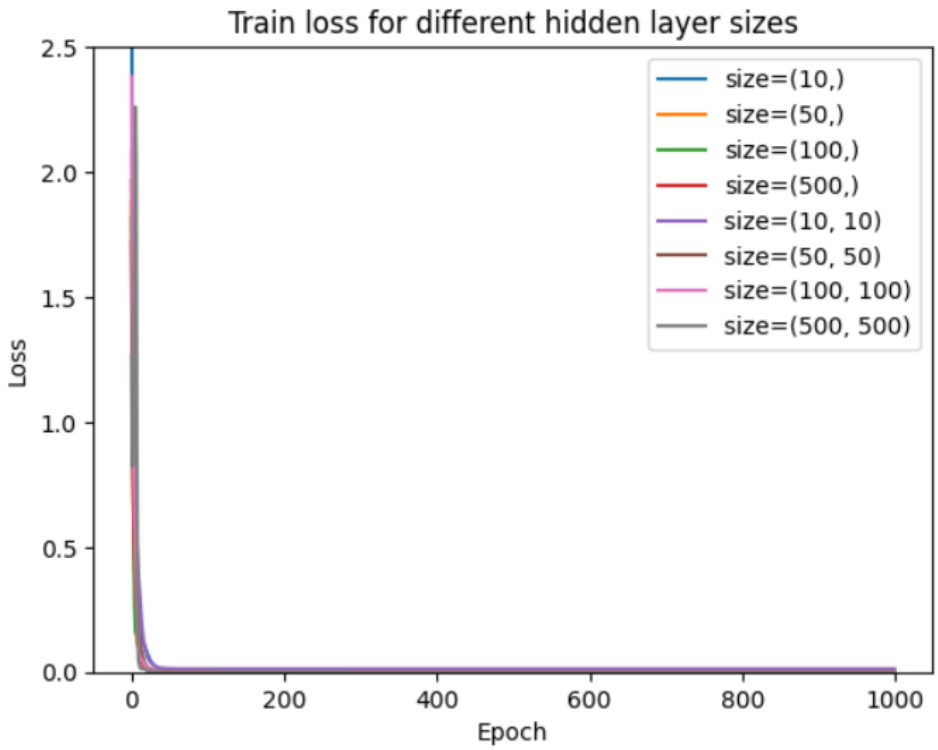


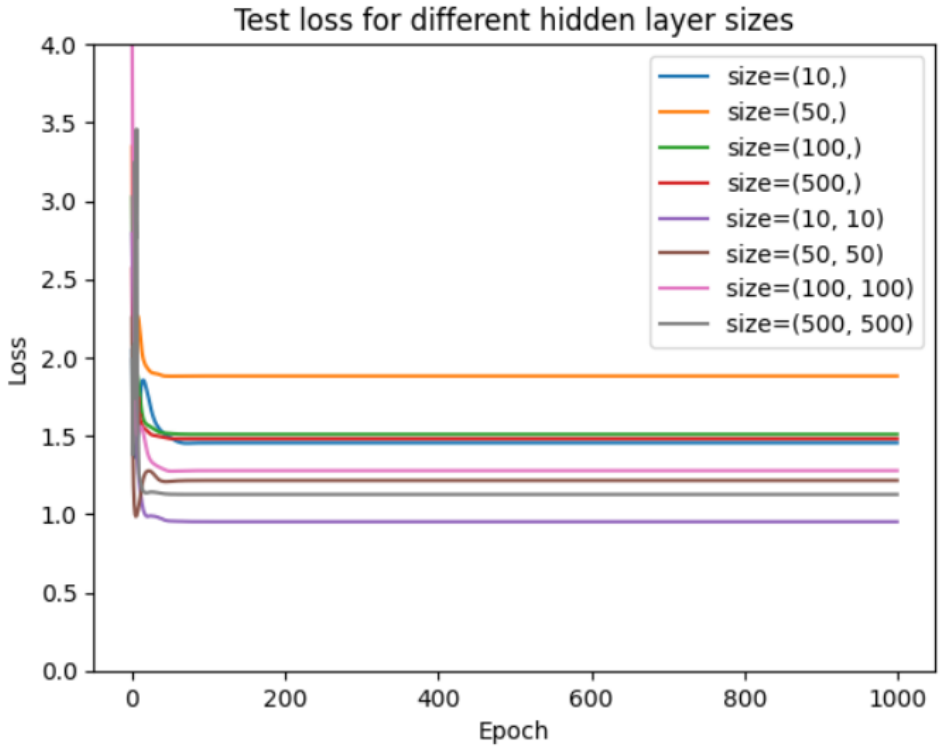
Wnioski z tej części eksperymentu:

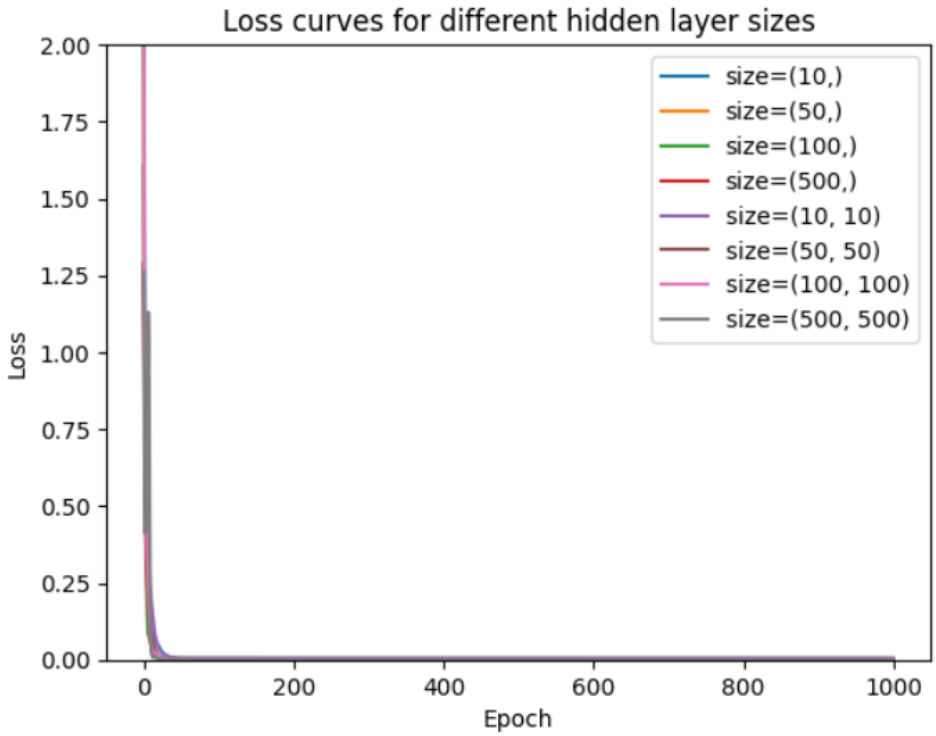
- learning_rate:
  - constant i adaptive – niemalże te same wyniki, model szybko popada w stagnację, jest słabo dopasowany. Ze względu na specyfikę tych dwóch ustawień za najlepsze dla mojego modelu uznaję tu ‘adaptive’
  - invscaling – największe wartości strat, tempo uczenia tak niskie, że model często wciąż nie skończył procesu uczenia się po 1000-cu epok, podczas gdy ‘constant’ i ‘adaptive’ dochodziły do stagnacji nie później niż na 400-tnej epoce
- learning_rate_init:
  - 0.0001 – zbyt niskie tempo uczenia, co widać szczególnie mocno dla ‘invscaling’
  - 0.02 – tempo uczenia nie wydaje się być wiele lepsze od innych opcji, wartości strat – choć mniejsze – to niewiele (różnice ledwo zauważalne, zwłaszcza przy ‘constant’/’adaptive’), dlatego ze względów ostrożności, aby nie wybrać wartości zbyt dużej, która mogłaby powodować niestabilności i trudności w osiągnięciu optymalnego minimum funkcji kosztu, za najlepsze ustawienie dla mojego modelu uznaję tu 0.01

Model wciąż wydaje się być słabo dopasowany do zestawu danych, stagnacja krzywej uczenia osiągana jest zdecydowanie zbyt szybko.

- Badanie wpływu rozmiaru modelu MLP na otrzymane wyniki







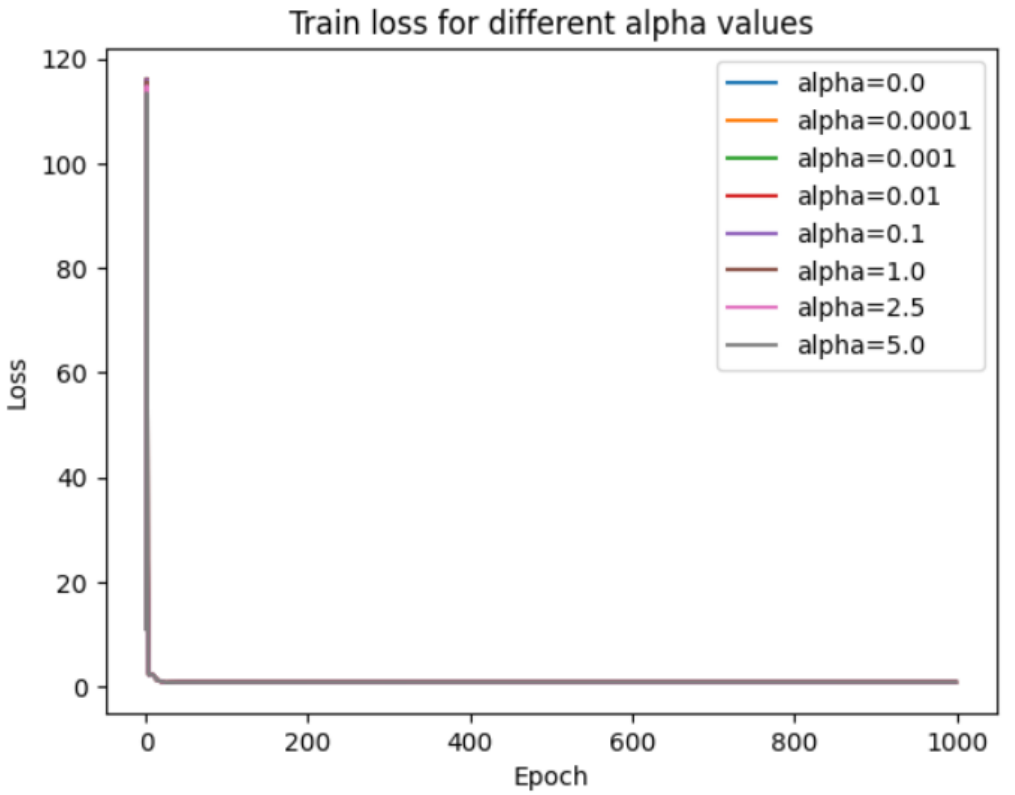
Wnioski z tej części eksperymentu:

Otrzymane wyniki są jednoznaczne – sieć zdecydowanie lepiej radzi sobie w przypadku wprowadzenia dwóch ukrytych warstw neuronów, w porównaniu do wyników otrzymywanych w przypadku tylko jednej warstwy ukrytej. Zaskakującym wynikiem jest jednak, że ze wszystkich kombinacji rozmiarów warstw najlepiej poradziły sobie dwie skrajne – (10, 10,) oraz (500, 500,). Sieć o rozmiarze (500,500,) jest zdecydowanie najbardziej skomplikowana ze wszystkich, przez co podejrzewam ją o to, że może być najbardziej podatna do overfittingu, natomiast sieć (10, 10,) jest bardzo mało skomplikowana, co mogłoby skutkować wystąpieniem underfittingu. Trzecie najmniejsze wartości strat dla zbioru testowego zostały osiągnięte przy sieci o rozmiarze (50, 50,), w przypadku tej krzywej dla zbioru testowego można jednak niestety zauważyć niepokojący skok na początku krzywej, przez co zdecydowałem, że dobrą opcją dla mojego modelu będzie rozmiar sieci (100, 100,) – nie jest to krzywa o najmniejszej wartości strat, nie jest ona jednak ani zbyt bardzo, ani zbyt mało skomplikowany rozmiar sieci i jego krzywa nie wykazuje żadnych niepokojących nieregularności jak w przypadku rozmiaru (50, 50,). Model niestety wciąż pozostaje słabo dopasowany do danych.

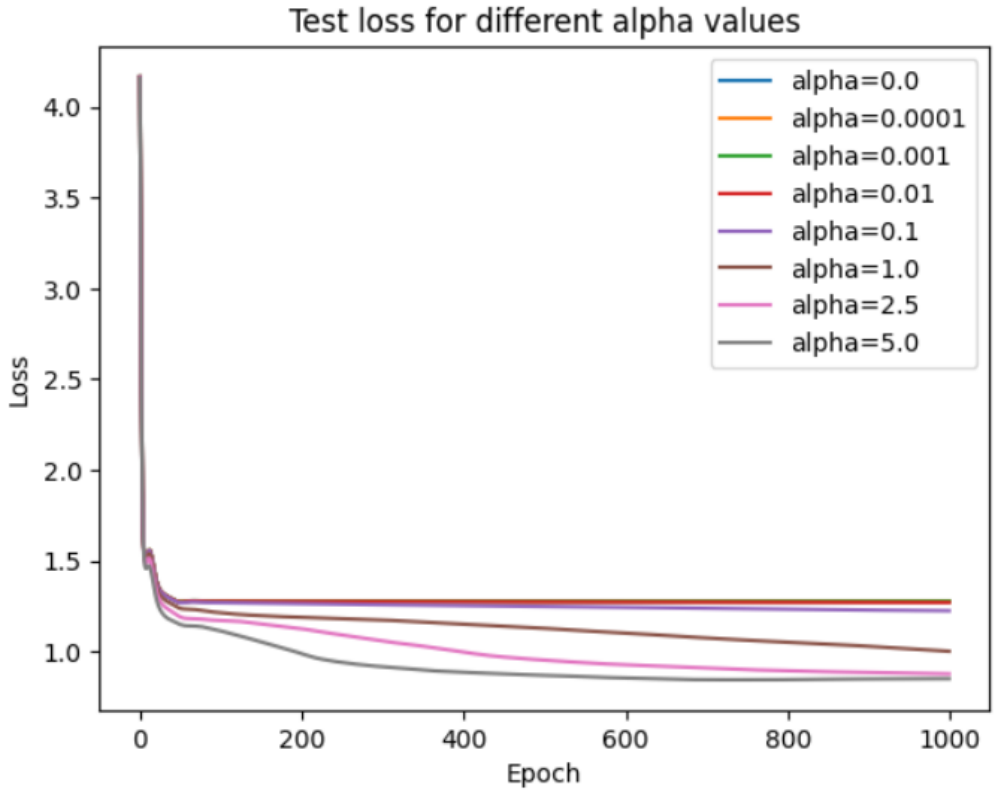
- Badanie wpływu regularyzacji na otrzymane wyniki

Badanie to wykonane zostało dla dwóch przypadków – pierwszy dla rozmiaru sieci (2000, 2000,), który jest znacznie większy od jakichkolwiek wcześniej testowanych wartości, w ramach próby wymuszenia overfittingu, a drugi (100, 100,) – dla porównania, jako że poprzednio uznałem ten rozmiar sieci za najbardziej odpowiedni. Próba overfittingu została podjęta ze względu na to, że regularyzacja jest techniką używaną do zapobiegania przeuczeniu modelu przez kontrolowanie jego złożoności.

BŁĘDNA ETYKIETA - wykres prezentuje krzwą strat dla danych testowych | Rozmiar sieci: (2000, 2000,)



Rozmiar sieci: (100, 100,)



Jak widać po przedstawionych powyżej wykresach, modulacje parametru ‘alpha’ dla sieci o rozmiarze (2000, 2000,) zakończyło się niepowodzeniem – zmiany są niewidoczne. Można je natomiast dostrzec wyraźnie w przypadku rozmiaru (100, 100,), gdzie różnice są w pełni zauważalne. Na pierwszy rzut oka

wydaje się, że najlepiej poradziła sobie wartość ‘alpha’=5.0, przy dokładnej obserwacji można zauważyć jednak, że około 650-tej epoki wartości krzywej strat zaczynają rosnąć, co może wskazywać na wystąpienie zjawiska przeuczenia – z tego powodu za najlepszą wartość regularyzacji uznaję 2.5 – w tym wypadku wartości krzywej strat są bardzo nieznacznie większe, widoczne jest jednak, że trend krzywej w tym przypadku jest spadkowy wedle moich oczekiwań.

- Testowanie najlepszej konfiguracji modelu MLP na własnych dowcipach

Ostateczna konfiguracja modelu:

```
best_model = MLPRegressor(
    solver='sgd',
    random_state=3,
    alpha=2.5,
    hidden_layer_sizes=(100,100,),
    learning_rate='adaptive',
    learning_rate_init=0.01,
    max_iter=1000,
)
best_model.fit(x_train_scaled, y_train)
```

✓ 3.2s

Metoda oceniająca śmieszność żartu:

```
def rate_my_joke(joke: str):
    embeeded_joke = model.encode([joke])
    embeeded_joke = np.reshape(embeeded_joke, (1, -1))
    #print(embeeded_joke.shape)

    prediction = best_model.predict(embeeded_joke)
    print(f'\nMy joke: {joke}\nMy joke rating on a scale [-10, 10]: {prediction}')
```

✓ 0.0s

Output:

```
Max mean rating: 3.3625929395733083
Min mean rating: -3.704540991199629

My joke: What vegetable should you never take on boat with you ? A leek !
My joke rating on a scale [-10, 10]: [-0.03373066]

My joke: Why was six afraid of seven? Because 7-8-9.
My joke rating on a scale [-10, 10]: [-0.09925713]

My joke: Whats the best thing about Switzerland? I dont know, but the flag is a big plus.
My joke rating on a scale [-10, 10]: [0.01809872]

My joke: Why don't scientists trust stairs? Because they're always changing steps and trying to elevate
their game!
My joke rating on a scale [-10, 10]: [0.09342576]

My joke: My brother came back from school all motivated because he said he would be following a new diet
from that day. We didn't really give it much thought until my brother really started eating his homework
for dinner. When we stopped him and asked why he was doing that, he replied: I was just trying to see how
it tasted because my teacher said that the homework would be a piece of cake for me.
```

My joke rating on a scale [-10, 10]: [0.24448499]

My joke: A businessman went into the office and found an inexperienced handyman painting the walls. The handyman was wearing two heavy parkas on a hot summer day. Thinking this was a little strange, the businessman asked the handyman why he was wearing the parkas on such a hot day. The handyman showed him the instructions on the can of paint. They read: “For best results, put on two coats.”
My joke rating on a scale [-10, 10]: [0.33915633]

My joke: How does dry skin affect you at work? You dont have any elbow grease to put into it.
My joke rating on a scale [-10, 10]: [0.16315717]

My joke: Why don't scientists trust atoms? Well, it all started with a group of scientists who were trying to understand the nature of matter. They were fascinated by atoms and their fundamental role in shaping the world around us. But as they delved deeper into their research, they started to notice something peculiar. Atoms seemed to have a mischievous side. Every time the scientists thought they had figured out the behavior of an atom, it would surprise them with new and unexpected properties. Atoms would bond and form molecules, create complex structures, and exhibit behaviors that seemed to defy conventional wisdom. The scientists realized that they couldn't fully trust atoms because they were constantly changing the rules. Just when they thought they had a grasp on how atoms worked, they would discover a new phenomenon that challenged their understanding. Despite the challenges, the scientists persevered. They continued to study atoms, unlock their secrets, and make groundbreaking discoveries. While they couldn't fully trust atoms, they couldn't deny their incredible power and influence in shaping the world we live in. so, the next time you hear someone say that scientists don't trust atoms, remember that it's not because they dislike them. It's because atoms are always keeping scientists on their toes, pushing the boundaries of knowledge and opening up new possibilities.
My joke rating on a scale [-10, 10]: [0.508611]

My joke: I just finished my last assignment for this course
My joke rating on a scale [-10, 10]: [0.03224067]