**Лабораторна робота №8**

**РЕСУРСИ KERAS. TENSORFLOW. НАВЧАННЯ ЛІНІЙНОЇ РЕГРЕСІЇ.**

**Мета роботи:** Дослідження ресурсу Keras і TensorFlow. Застосування TensorFlow.

Хід роботи:

**Завдання 8.1** Кластеризація даних за допомогою методу k-середніх

Лістинг коду: (TensorFlow 1.x більше не підтримується, а в TensorFlow 2.x були вилучені функції, які були в першій версії, тому код модернізований)

import numpy as np  
import tensorflow as tf  
  
# Параметри  
n\_samples, batch\_size, num\_steps = 1000, 100, 20000  
  
# Генеруємо дані  
X\_data = np.random.uniform(1, 10, (n\_samples, 1)).astype(np.float32)  
y\_data = (2 \* X\_data + 1 + np.random.normal(0, 2, (n\_samples, 1))).astype(np.float32)  
  
# Ініціалізуємо параметри  
k = tf.Variable(tf.random.normal((1, 1)), name='slope')  
b = tf.Variable(tf.zeros((1,)), name='bias')  
  
# Лінійна регресія  
def model(X):  
 return tf.matmul(X, k) + b  
  
# Функція втрат (MSE)  
def compute\_loss(y\_true, y\_pred):  
 return tf.reduce\_mean(tf.square(y\_true - y\_pred))  
  
# Оптимізатор  
optimizer = tf.optimizers.SGD(learning\_rate=0.01)  
  
# Навчальний цикл  
display\_step = 100

for i in range(num\_steps):  
 indices = np.random.choice(n\_samples, batch\_size)  
 X\_batch, y\_batch = X\_data[indices], y\_data[indices]  
  
 # Обчислення градієнтів та оптимізація  
 with tf.GradientTape() as tape:  
 y\_pred = model(X\_batch)  
 loss = compute\_loss(y\_batch, y\_pred)

gradients = tape.gradient(loss, [k, b])  
 optimizer.apply\_gradients(zip(gradients, [k, b]))

# Відображаємо прогрес  
if (i + 1) % display\_step == 0:  
 print(f'Похибка {i + 1}: {loss:.8f}, k={k.numpy()[0][0]:.4f}, b={b.numpy()[0]:.4f}')

Результат виконання програми:

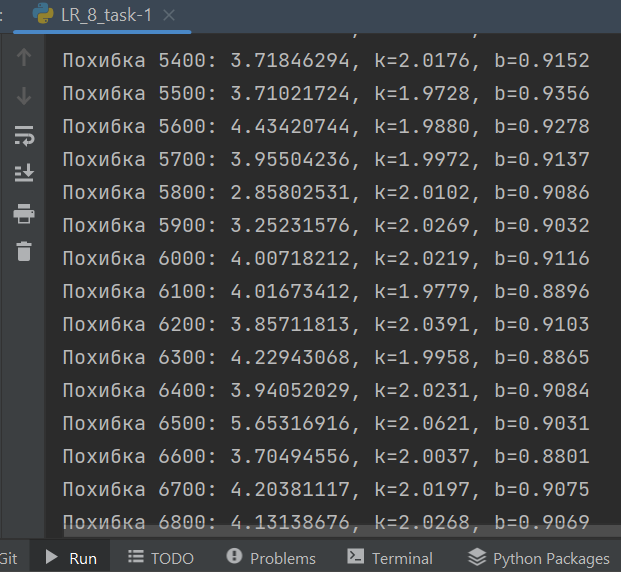


Рисунок 1 - Результат виконання програми

### 1. **Похибка (loss)**:

* **Похибка** вказує, наскільки **передбачені значення** моделі (y\_pred) відрізняються від **реальних значень** (y).
* Наприклад, на кроці **5800** похибка становить **2.85**, а на кроці **6400** — **3.94**.
* Похибка не зменшується монотонно, оскільки в алгоритмі використовуються **випадкові батчі даних** під час кожної ітерації. Це викликає коливання.

### 2. **Параметр k (схил)**:

* **k** — це коефіцієнт (нахил прямої), який модель намагається навчити.
* В ідеальному випадку (без шуму у даних) **k ≈ 2**, оскільки моя модель була задана як y = 2 \* X + 1 + noise.
* З результатів видно, що **k** наближається до **2**, але трохи коливається навколо нього:
  + Наприклад: k=2.0102, k=2.0269, k=2.0391.

### 3. **Параметр b (зміщення)**:

* **b** — це константа зміщення (intercept), яку також потрібно навчити.
* В ідеальному випадку **b ≈ 1**.
* У результатах значення **b** також близьке до **1**, але трохи відхиляється через **шум** у даних та стохастичність навчання:
  + Наприклад: b=0.9086, b=0.9116, b=0.8865.

**Висновок:** дослідив ресурси Keras і TensorFlow. Застосував TensorFlow.