LeetCode difficulty estimator

Neural networks semestral project

Mihal Filip; Trappl Juraj; 2024

LeetCode

- popular platform for developers
- rich community
- content
    - Contests
    - Mock interviews for jobs
    - Problems/Tasks
    - Forum discussions
    - Courses

Our task

- Given a description of a problem, predict its difficulty
    - Easy / Medium / Hard

1. Two Sum

Easy 54151 1803 Add to List Share

Difficulty (target)

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

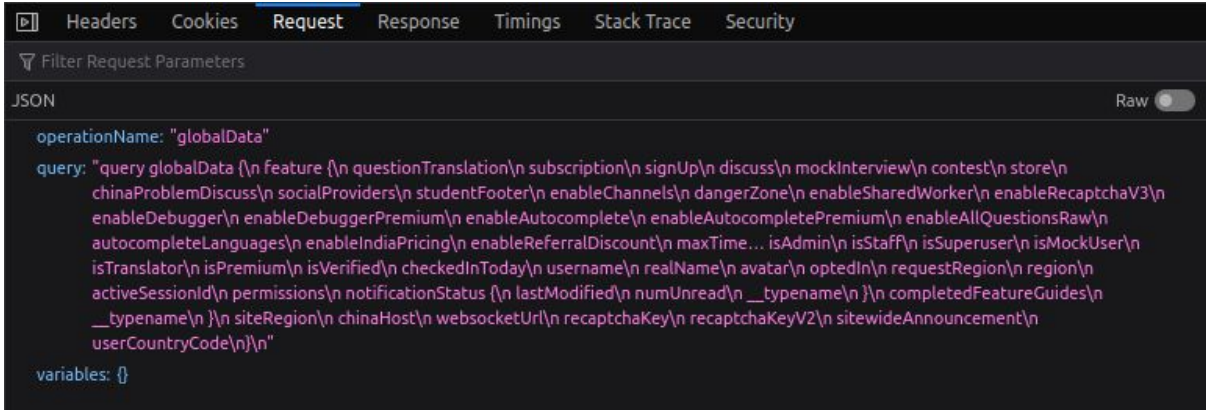
You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Textual description (data)

How to obtain the data?

- Query the LeetCode internal API



GraphQL

Querying the API

```
graphql_request_problems_list = leetcode.GraphqlQuery(
    query="""
        query problemsetQuestionList($categorySlug: String, $limit: Int, $skip: Int, $filters: QuestionListFilterInput) {
            problemsetQuestionList: questionList(categorySlug: $categorySlug, limit: $limit, skip: $skip, filters: $filters) {
                total: totalNum
                questions: data {
                    acRate
                    difficulty
                    title
                    titleSlug
                    topicTags {
                        name
                        id
                        slug
                    }
                    hasSolution
                }
            }
        }
    """,
    variables=variables,
    operation_name="problemsetQuestionList"
)
```

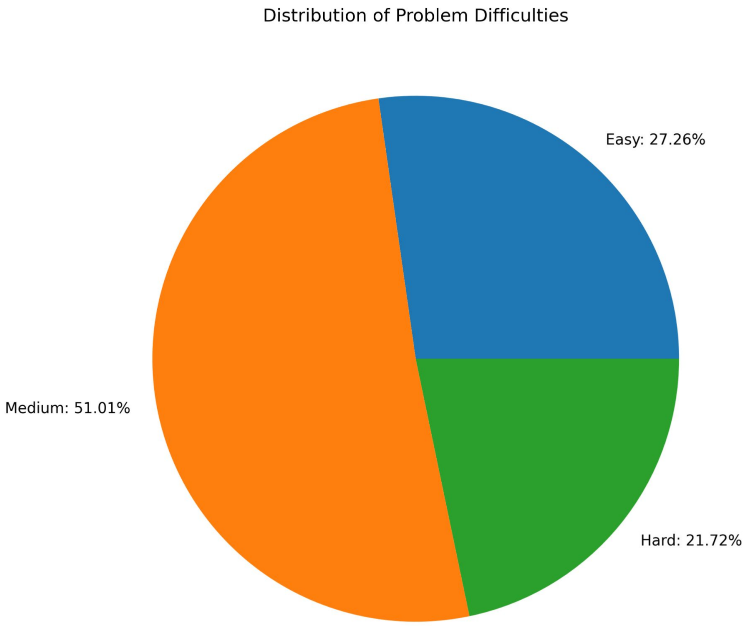
1. Query the list of problems names

```
# For each problem, query the difficulty and description.
result = {}
for title_slug in problems:
    variables = {
        "titleSlug": title_slug
    }

    graphql_request = leetcode.GraphqlQuery(
        query="""
            query questionData($titleSlug: String!) {
                question(titleSlug: $titleSlug) {
                    content
                    difficulty
                }
            }
        """,
        variables=variables,
        operation_name="questionData",
    )

    api_response = api_instance.graphql_post(body=graphql_request).to_dict()["data"]["question"]
    result[title_slug] = {
        "difficulty": api_response["difficulty"],
        "content": api_response["content"]
    }
```

2. For each problem query the data

Dataset

- 2950 problems
  - ~584 problems for premium users
- **2366** examples to train



Understanding training examples

Lots of HTML clutter


```
        "content": "<p>Given an array of integers <code>nums</code> and an integer <code>target</code>, return
<em>indices of the two numbers such that they add up to <code>target</code></em>.</p>\n\n<p>You may assume that each
input would have <strong><em>exactly</em> one solution</strong>, and you may not use the <em>same</em> element twice.
</p>\n\n<p>You can return the answer in any order.</p>\n\n<p> </p>\n<p><strong class=\"example\">Example
1:</strong></p>\n\n<pre>\n<strong>Input:</strong> nums = [2,7,11,15], target = 9\n<strong>Output:</strong>
[0,1]\n<strong>Explanation:</strong> Because nums[0] + nums[1] == 9, we return [0, 1].\n</pre>\n\n<p><strong
class=\"example\">Example 2:</strong></p>\n\n<pre>\n<strong>Input:</strong> nums = [3,2,4], target = 6\n<strong>Output:
</strong> [1,2]\n</pre>\n\n<p><strong class=\"example\">Example 3:</strong></p>\n\n<pre>\n<strong>Input:</strong> nums
= [3,3], target = 6\n<strong>Output:</strong> [0,1]\n</pre>\n\n<p> </p>\n<p><strong>Constraints:</strong></p>\n
\n<ul>\n\t<li><code>2 &lt;= nums.length &lt;= 10<sup>4</sup></code></li>\n\t<li><code>-10<sup>9</sup> &lt;= nums[i]
&lt;= 10<sup>9</sup></code></li>\n\t<li><code>-10<sup>9</sup> &lt;= target &lt;= 10<sup>9</sup></code></li>\n\t<li>
<strong>Only one valid answer exists.</strong></li>\n</ul>\n\n<p> </p>\n<strong>Follow-up: </strong>Can you
come up with an algorithm that is less than <code>O(n<sup>2</sup>)</code><font face=\"monospace\"> </font>time
complexity?"
```


Sometimes there are examples mentioned

Sometimes required complexity is mentioned

- What is relevant? What should be considered as features?

Feature extraction

- Tf-Idf
    - Word-level & character-level ngrams
- Word embeddings
    - Static
        - GloVe
        - fastText
    - Contextualized
        - BERT

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Shallow learning models

- 80:20 train:test
- Tf-idf
    - ~400k features (1-3 word-level, 1-5 character level)
- class imbalance problem
    - Oversampling (SMOTE)
    - balanced batches
    - **downsampling**

Classification

(1 out of 3 classes)

- Perceptron
- SVM linear kernel
- MLP Classifier

Regression

(prediction on a scale from 0 to 2)

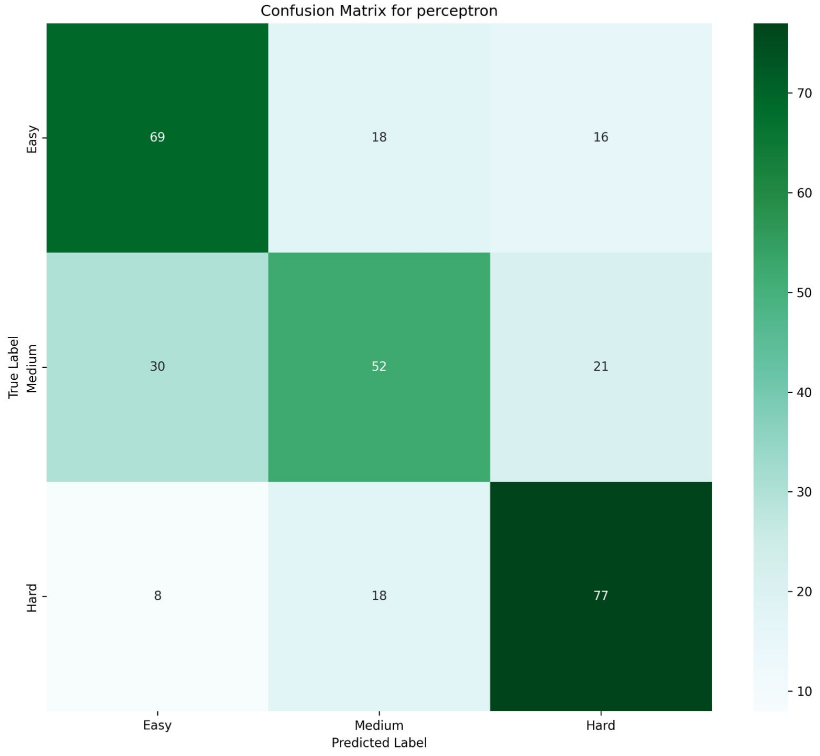
- MLP Regressor

Perceptron

Best model:

- Tf-idf
    - 1 to 5 character level n-grams
    - 1 to 3 word level n-grams
- SelectPercentile(percentile=50)
- *L2* penalty
- alpha 0.0001

Stratified k-fold, k = 5
Average test macro f1: **58.26**
Average test micro f1: **58.56**

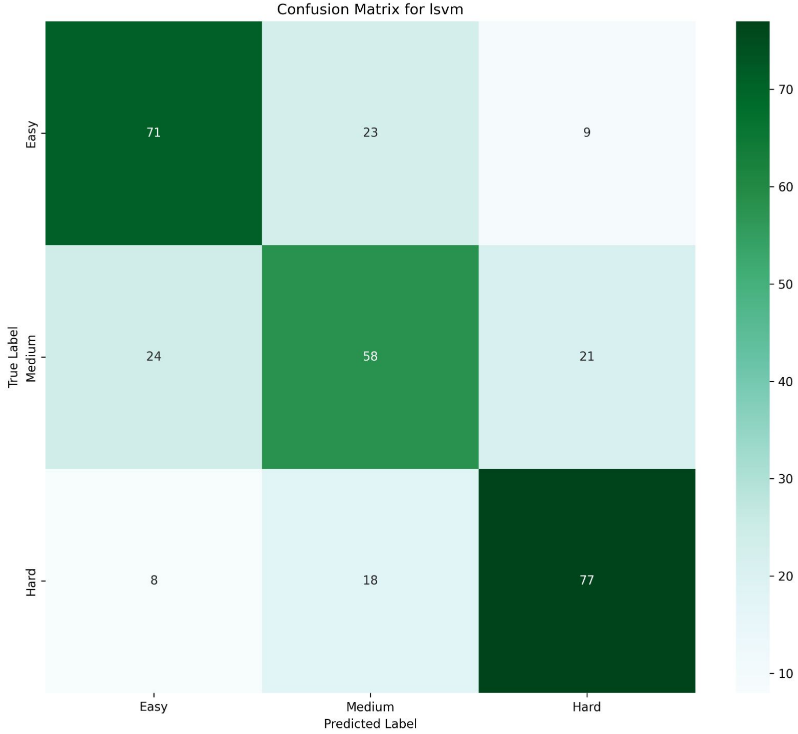


Linear SVM

Best model:

- Tf-idf
  - 1 to 5 character level n-grams
  - 1 to 3 word level n-grams
- SelectPercentile(percentile=5)
- Square hinge loss
- *L2* penalty
- alpha 0.0001

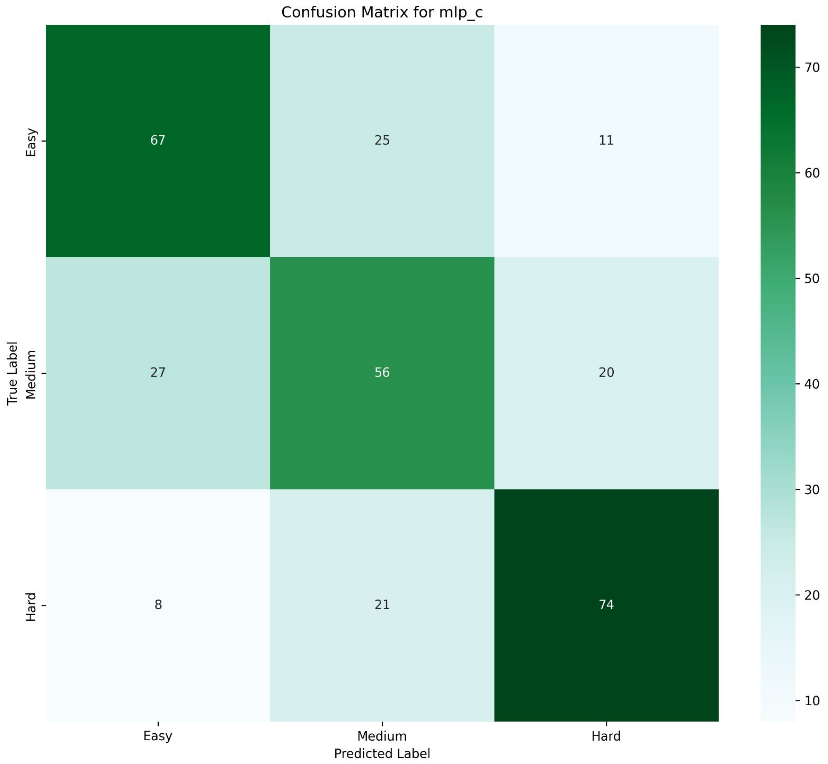
Stratified k-fold, k = 5
Average test macro f1: **60.94**
Average test micro f1: **61.15**



MLP Classifier

- Adam optimizer, momentum 0.9 with Nesterov, ReLU, Tf-Idf word & character 3-grams, 5-grams
- Stratified k-fold, k = 5
- SelectPercentile(percentile=5)

Hidden layer(s) size	Average test macro f1	Average test micro f1
(64,)	59.42	59.61



MLP regressor

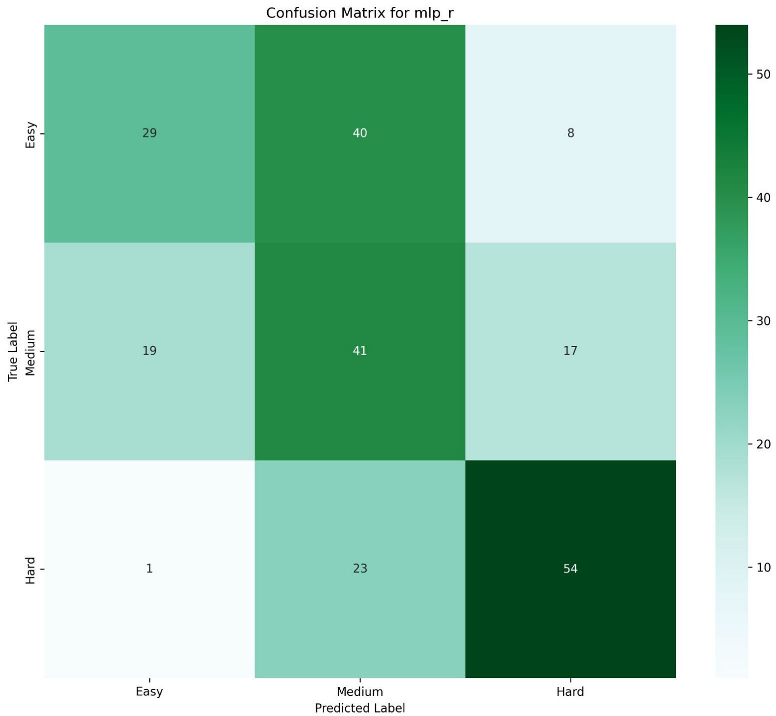
linear scale

0 - easy, 1 - medium, 2-hard

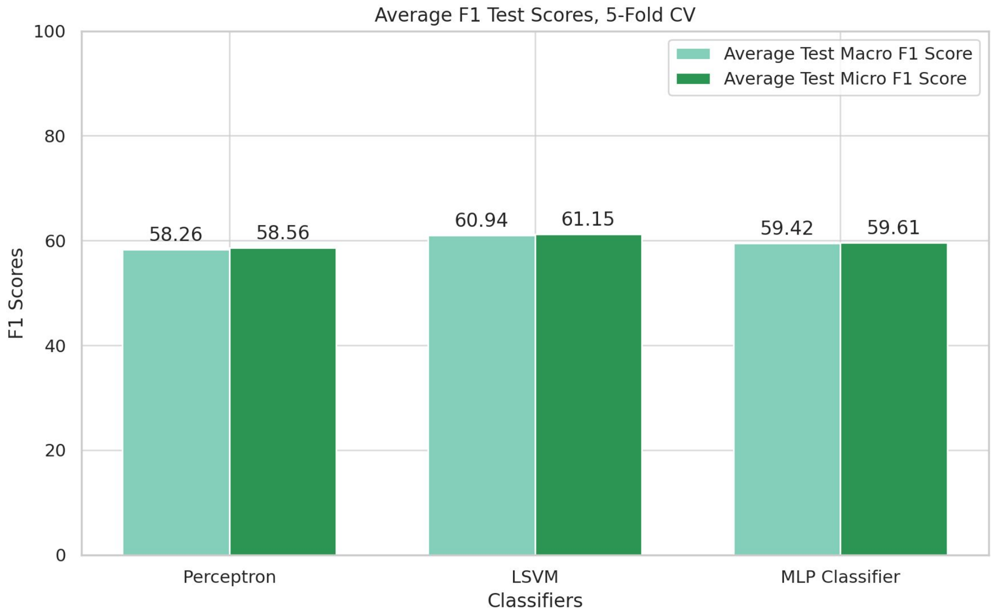
rounding the predictions 0.75, 1.25 thresholds

Test RMSE: **0.675**

Test accuracy: **53.4%**



Shallow learning summary



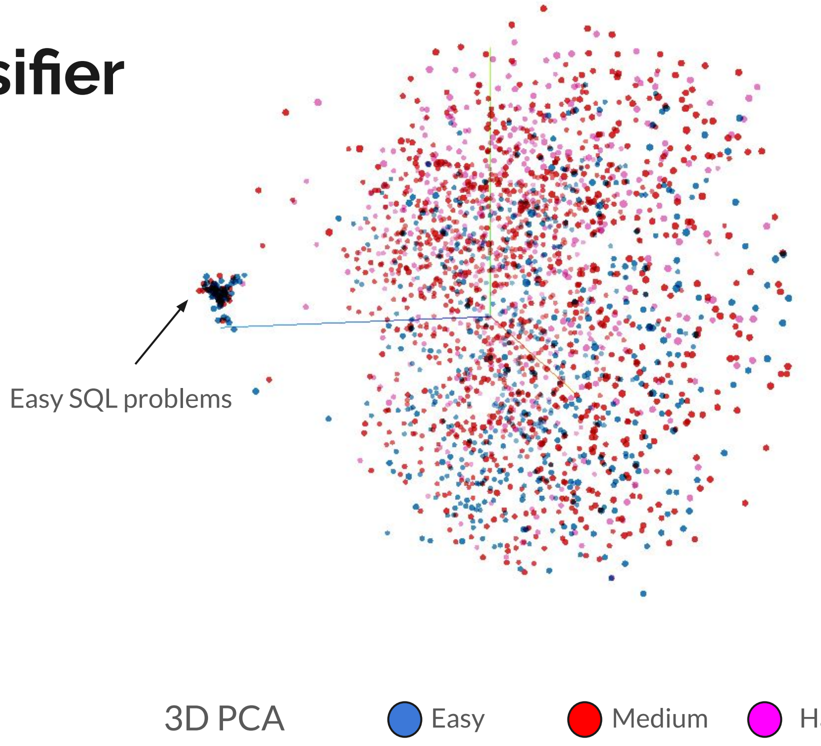
Classification using contextualized embeddings

- Tensorflow 2.12, KerasTuner
    - Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization
- Pre-trained bert-base-uncased
    - No fine-tuning
- General training information
    - Batch size 32
    - Datasets sizes 1079:231:232 samples (70:15:15)
- On average, contextualized representations are more context-specific in higher layers
    - https://ai.stanford.edu/blog/contextual/

1. layer BERT dense classifier

Features

- 1. Layer BERT embeddings
- mean pooling
- (768,)

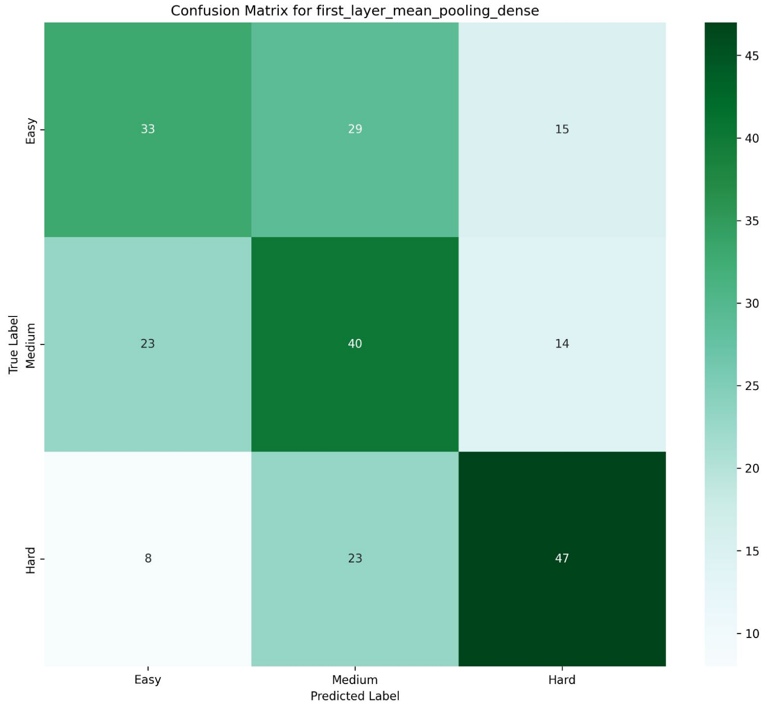


1. layer BERT dense classifier

Model

- Classification head
    - 1 x Dense
        - 576 (relu)
    - Adam optimizer
    - no Batch Norm
    - initial lr 0.01 with cosine decay
    - dropout 0.4
    - L1 ~0.00004
    - L2 ~0.0003

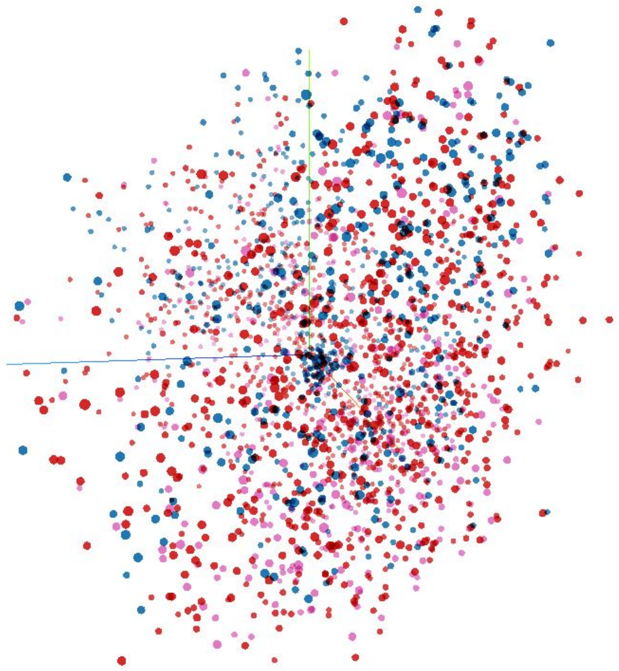
Test accuracy: **51.7%**



2. layer BERT dense classifier

Features

- 2. Layer BERT embeddings
- mean pooling
- (768,)



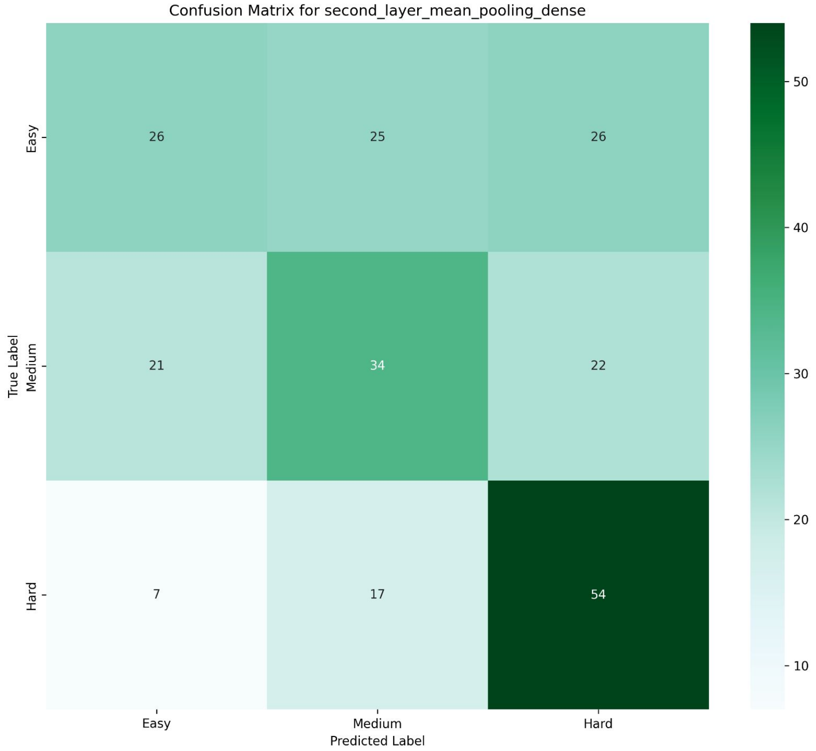
3D PCA Easy Medium Hard

2. layer BERT dense classifier

Model

- Classification head
  - 1 x Dense
    - 960 (selu)
  - SGD optimizer, momentum 0.9 with Nesterov
  - no Batch Norm
  - initial lr 0.01 with cosine decay
  - dropout 0.1
  - L1 ~2.25
  - L2 ~1.27

Test accuracy: **49.1%**



In-context learning classification

- *The ability of to learn from the context provided in the prompt*
- Few-shot learning
- Llama-2-13b-chat-hf
    - pretrained and fine-tuned generative text model
    - optimized for dialogue
    - `13 billion parameters * 32 bits/parameter ~ 52 GB`

PyTorch

LangChain

HUGGING FACE

Prompt

- One example of each problem
  - Q3 acceptance rate

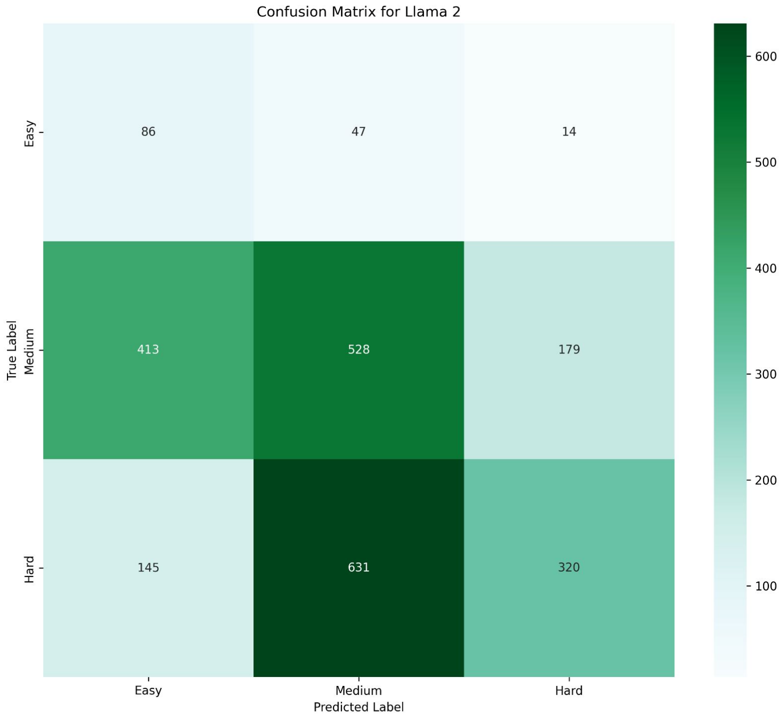
Types of solutions to example problems:

- Easy
  - Single loop solution
- Medium
  - DP
- Hard
  - Pure math

```
prompt = PromptTemplate.from_template(
    """
    <s>[INST] <<SYS>>
    Task: Given a programming problem description, predict its difficulty.
    The difficulty can be one of easy, medium and hard.

    Example:
    Given a programming problem description: {easy_programming_problem_example}, the difficulty is:
    easy

    Example:
    Given a programming problem description: {medium_programming_problem_example}, the difficulty is:
    medium

    Example:
    Given a programming problem description: {hard_programming_problem_example}, the difficulty is:
    hard

    <<SYS>>
    Now, given a programming problem description: {programming_problem}, the difficulty is:
    [/INST]
    """
)
```

Llama 2 results

- Trained on 3 examples
- Test set was the rest
  - 2360 problems
- Accuracy **39.65%**



Improvements

- Custom tokenization
  - E.g. handle mathematical expressions
- Fine-tune LLM first, then extract embeddings
- Try other models like ELMo, GPT-2 for comparison

Showcase

You are given an integer n. Return the sum of first n fibonacci numbers.

Constraints:

1 <= n <= 100

Best model prediction: **Easy**

You are given a hard task. Compute average of 2 numbers a and b.

Best model prediction: **Easy**

I am losing my interest in human beings; in the significance of their lives and their actions. Someone has said it is better to study one man than ten books. I want neither books nor men; they make me suffer. Can one of them talk to me like the night – the Summer night? Like the stars or the caressing wind? The night came slowly, softly, as I lay out there under the maple tree. It came creeping, creeping stealthily out of the valley, thinking I did not notice. And the outlines of trees and foliage nearby blended in one black mass and the night came stealing out from them, too, and from the east and west, until the only light was in the sky, filtering through the maple leaves and a star looking down through every cranny.

<u>You are given an integer n. Return the sum of first n fibonacci numbers.</u>
<u>Constraints:</u>
<u>1 <= n <= 100</u>

Best model prediction: **Medium**

Inference problems

- Difficulty may seem subjective
    - Easy vs. Medium / Medium vs. Hard?
- We want to minimize Easy vs. Hard problems
    - Easy problems predicted being Hard
    - Hard problems predicted being Easy
- An example of a common problem
    - Hard to classify “min-max” problems correctly

2078. Two Furthest Houses With Different Colors

Easy 865 24 Add to List Share

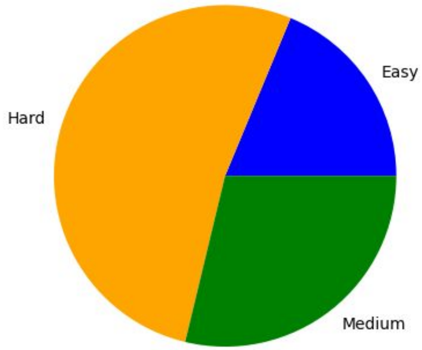
There are `n` houses evenly lined up on the street, and each house is beautifully painted. You are given a **0-indexed** integer array `colors` of length `n`, where `colors[i]` represents the color of the i^{th} house.

Return *the **maximum** distance between **two** houses with **different** colors.*

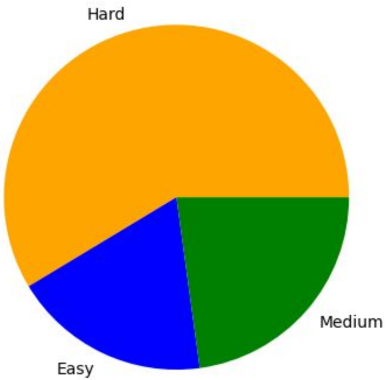
The distance between the i^{th} and j^{th} houses is `abs(i - j)`, where `abs(x)` is the **absolute value** of `x`.

Predicted as hard by each classifier.

Distribution of difficulties of problems that contain "maximum"



Distribution of difficulties of problems that contain "minimum"



Thank you for your attention!