
ML 2025 Spring HW2

TAs: 林毓翔 李晨安 上官世昀

Email: ntu-ml-2025-spring-ta@googlegroups.com

Deadline: 2025/3/28 23:59:59 (UTC+8)

Links

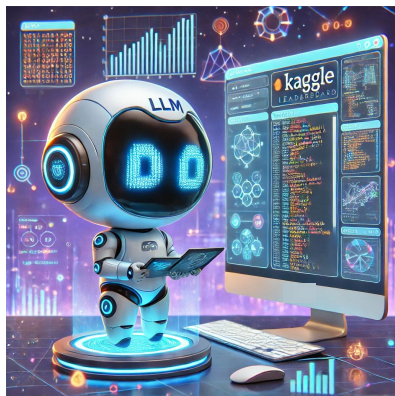
NTU COOL	https://cool.ntu.edu.tw/courses/46406/discussion_topics/375678
Colab Sample Code	https://colab.research.google.com/drive/1NmccKSzfQwf4m1Rzjb_kCXOup7hWbpsT?usp=sharing
Kagge Sample Code	https://www.kaggle.com/code/yuxianglin032/ml2025spring-hw2-public
Kagge Invitation Link	https://www.kaggle.com/t/486f3f5988bd4240b6476b08ab579d0a
Kaggle Competition	https://www.kaggle.com/competitions/ml-2025-spring-hw-2

Outline

- Task Introduction
- Code Overview
- Grading
- Hints
- FAQs

Task Introduction - AI Agent As a Data Scientist

- Use LLMs to automatically write code
- By only knowing the dataset, LLMs can generate customized solutions for each task.



Source: ChatGPT

Task - Disease Prediction

- Given survey results in the past 3 days in a specific state in U.S., then predict the percentage of new tested positive cases in the 3rd day.
- Source: Delphi group @ CMU
- Do NOT search for or use additional data for training or the answers for the testing data.
- The LLM agent serves as your representative—if it violates the rules, it's as if you did.

One-hot Encoding

Dataset - Features (1)

$$AL = [1 \ 0 \ 0]$$

$$AZ = [0 \ 1 \ 0]$$

$$CA = [0 \ 0 \ 1]$$

- States (35, encoded to one-hot vectors)

- COVID-like illness

cli, ili, wnohh_cmnty_cli, hh_cmnty_cli, nohh_cmnty_cli

- Behavior indicators

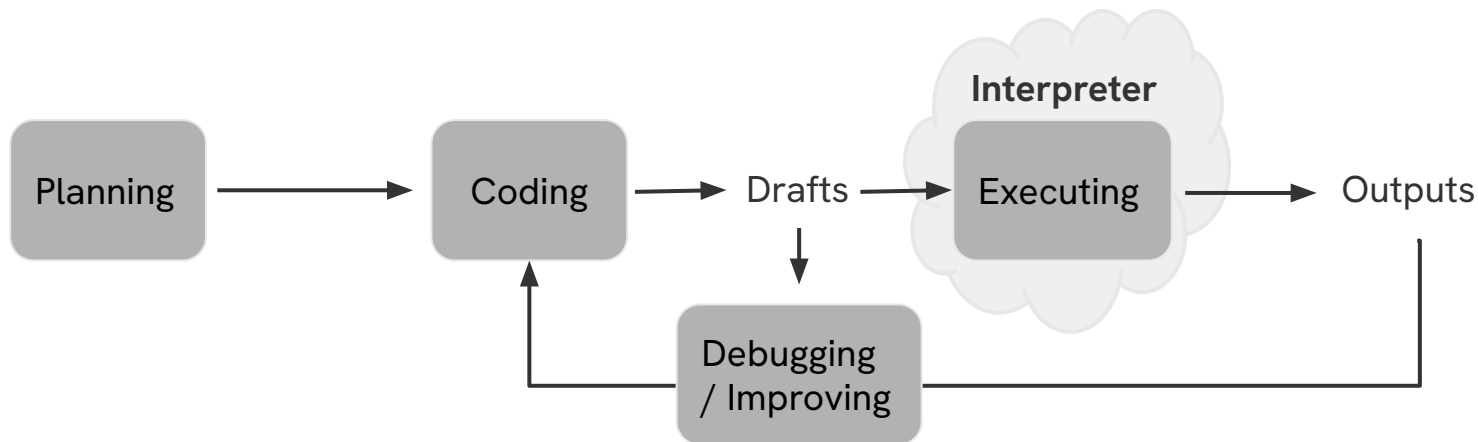
wearing_mask_7d, shop_indoors, restaurant_indoors, public_transfit, wlarge_event_indoors

Dataset - Features (2)

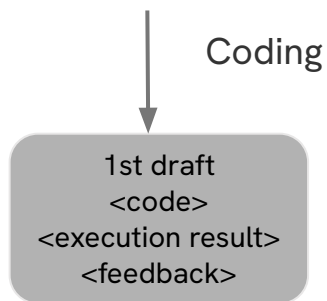
- ❑ Belief indicators
wbelief_mask_effective, wbelief_distancing_effective
- ❑ Mental indicator
wworried_catch_covid, wworried_finance
- ❑ Environmental indicators
**wother_masked_public, wother_distanced_public,
wcovid_vaccinated_friends**
- ❑ Tested Positive Cases
tested_positive

Sample Code Structure

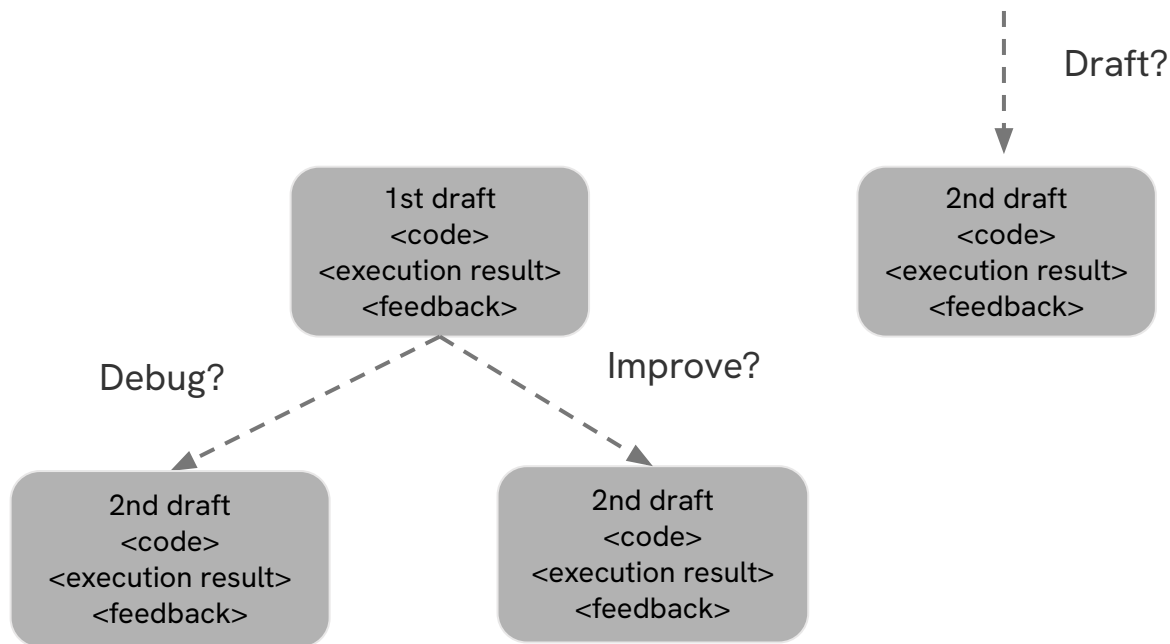
- The only thing we need to do is instruct LLM agents to handle that!



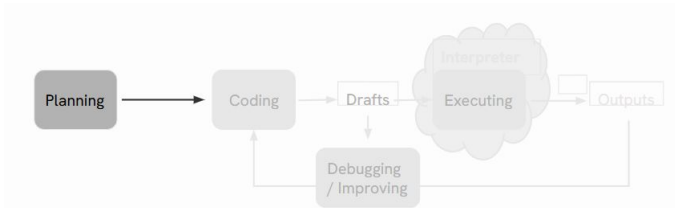
Action Space - First Step



Action Space - Second Step



Artifact Examples - Plan



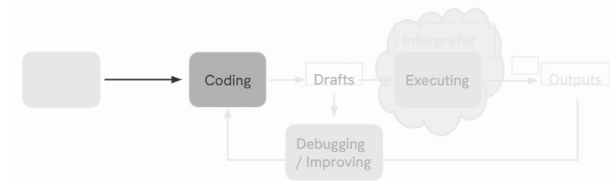
...

We will use the following steps:

1. Load the necessary libraries and data.
2. Preprocess the data by handling missing values and scaling the features.
3. Split the data into training and testing sets.
4. Train a regression model on the training data.
5. Make predictions on the testing data.
6. Save the predictions to a submission file.



Artifact Examples - Code

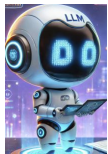


```
# Define the features and the target
X_train = train_df.drop(['id', 'tested_positive'], axis=1)
y_train = train_df['tested_positive']
X_test = test_df.drop(['id', 'tested_positive'], axis=1)

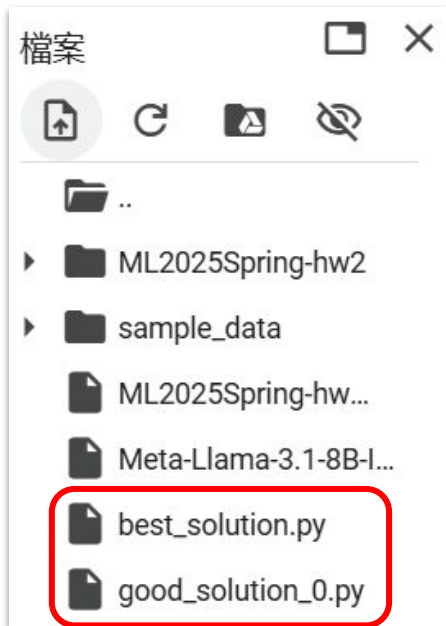
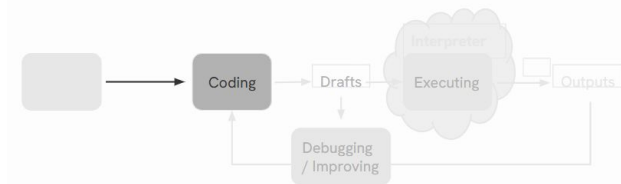
# Define the categorical and numerical features
categorical_features = X_train.select_dtypes(include=['object']).columns
numerical_features = X_train.select_dtypes(include=['int64', 'float64']).columns

# Define the preprocessing steps
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])
```



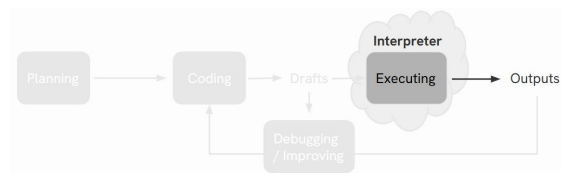
Artifact Examples - Code



```
# Define a function to save the best solution and other good solutions to files.
def save_run(cfg, journal):
    # Retrieve and save the best found solution.
    best_node = journal.get_best_node(only_good=False) # Get the best node.
    with open("best_solution.py", "w") as f:
        f.write(best_node.code)

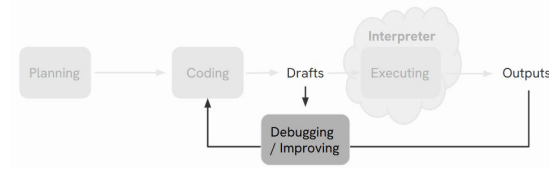
    good_nodes = journal.get_good_nodes() # Retrieve all good solution nodes.
    for i, node in enumerate(good_nodes):
        filename = f"good_solution_{i}.py"
        with open(filename, "w") as f:
            f.write(node.code)
```

Artifact Examples - Execution



```
Traceback (most recent call last):
  File "/content/best_solution.py", line 20, in <module>
    X_test = test_data.drop(['id', 'tested_positive_day3'], axis=1)
              ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py", line 5581, in drop
    return super().drop(
           ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/generic.py", line 4788, in drop
    obj = obj._drop_axis(labels, axis, level=level, errors=errors)
           ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/generic.py", line 4830, in _drop_axis
    new_axis = axis.drop(labels, errors=errors)
                ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py", line 7070, in drop
    raise KeyError(f"{labels[mask].tolist()} not found in axis")
KeyError: "['tested_positive_day3'] not found in axis"
```

Artifact Examples - Feedbacks



The error message indicates that the feature names seen at fit time are missing at predict time. This is because the feature names used during training are not the same as the ones used during prediction.

...

To fix this issue, you should scale the testing data using the same `StandardScaler` instance that was used during training. Here's how you can modify your code to do this:



Grading

Code Submission (+4 pts)	Submit your code to NTU COOL.
Public Simple Baseline (+1 pt)	Just run the provided sample code and submit 'submission.csv' to the Kaggle competition.
Private Simple Baseline (+1 pt)	
Public Medium Baseline (+1 pt)	Improve your LLM agents with the following methods: <ul style="list-style-type: none"><input type="checkbox"/> Prompt Engineering<input type="checkbox"/> Feature Selection<input type="checkbox"/> Use different LLM<input type="checkbox"/> Drafting More & Improving & Debugging<input type="checkbox"/> ...
Private Medium Baseline (+1 pt)	
Public Strong Baseline (+1 pt)	
Private Strong Baseline (+1 pt)	

Grading - Code Submission

- Submit to NTU COOL
- Deadline: 3/28 (Fri.) 23:59
- Compress your code into <student ID>_hw2.zip (e.g. b13901001_hw2.zip)
- We can only see your last submission.
- Do not submit model weights or dataset.
- If your code is not reasonable or reproducible, you will receive 0 points for this homework.

Grading - Baselines

Evaluation Metric


$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

prediction value

ground truth

Public Set

Estimated Time

#	Team	Score
	Strong Baseline	0.84773
	Medium Baseline	0.91179
	Simple Baseline	1.31311

20~ 30 minutes on T4 (Colab)

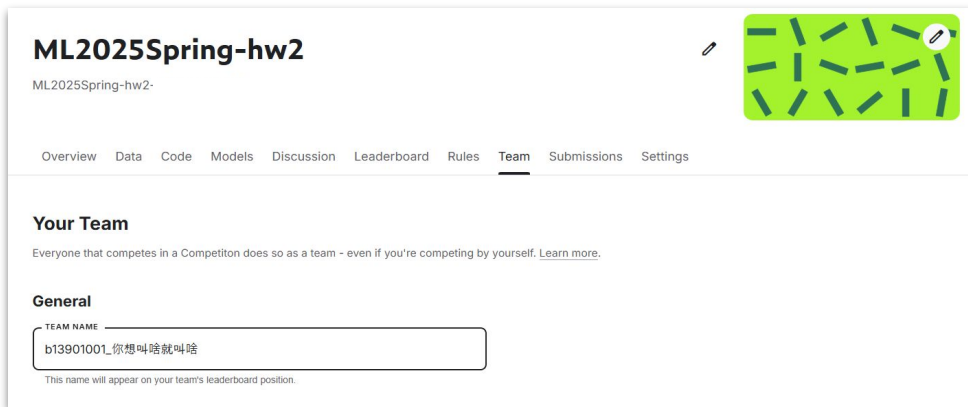
5 minutes on T4 (Colab)

5 minutes on T4 (Colab)

Passing public baselines doesn't guarantee that you will pass private ones.

Grading - Kaggle

- Before deadline, you can only see your public set performance
- Only 2 submission files will be scored on private set
- You can submit your prediction at most 5 times a day
- The team name must be in the format <student ID>_<anything>, or you will receive 0 point.



The screenshot shows the Kaggle competition interface for 'ML2025Spring-hw2'. The 'Team' tab is selected in the navigation bar. Under the 'Your Team' section, there is a text input field for the team name. The input field contains the text 'b13901001_你想叫啥就叫啥'. Below the input field, a small note states: 'This name will appear on your team's leaderboard position.'

Hints - Prompt Engineering

Imagine that you're teaching a student to code!

```
def _draft(self) -> Node:
```

```
# TODO: ask LLM agents to come up with a solution and then implement
```

```
system_prompt = "You are an AI agent."
```

```
user_prompt = [
```

```
    "You have to come up with a solution for machine learning task and then implement this solution in Python."
```

```
    f"The task is to {str(self.cfg.task_goal)} ",
```

```
    f'All the provided input data is stored in "{self.cfg.data_dir}" directory.',
```

```
    f"{str(self.data_preview)}",
```

```
    'You have to save the predictions result on testing set in "/content/submission.csv".'
```

```
]
```



Hints - Prompt Engineering

Imagine that you're teaching a student to code!

```
Traceback (most recent call last):
  File "/content/best_solution.py", line 20, in <module>
    X_test = test_data.drop(['id', 'tested_positive_day3'], axis=1)
              ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py", line 5581, in drop
    return super().drop(
           ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/generic.py", line 4788, in drop
    obj = obj._drop_axis(labels, axis, level=level, errors=errors)
           ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/generic.py", line 4830, in _drop_axis
    new_axis = axis.drop(labels, errors=errors)
                ~~~~~
  File "/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py", line 7070, in drop
    raise KeyError(f"{labels[mask].tolist()} not found in axis")
KeyError: "['tested_positive_day3'] not found in axis"
```

Hints - Prompt Engineering

Imagine that you're teaching a student to code!

```
def _draft(self) -> Node:

    # TODO: ask LLM agents to come up with a solution and then implement

    system_prompt = "You are an AI agent."

    user_prompt = [
        "You have to come up with a solution for machine learning task and then implement this solution in Python.",
        f"The task is to {str(self.cfg.task_goal)} ",
        f'All the provided input data is stored in "{self.cfg.data_dir}" directory.',
        f"{str(self.data_preview)}",
        'You have to save the predictions result on testing set in "/content/submission.csv".',
        'Note that the testing file DOES NOT have the target column.'
    ]
```

Hints - Feature Selection

When you have to make a prediction yourself, what information do you need?

```
def preview_csv(p: Path, file_name: str) -> str:
    """Generate a textual preview of a csv file"""

    df = pd.read_csv(p)

    out = []

    out.append(f"-> {file_name} has {df.shape[0]} rows and {df.shape[1]} columns.")

    # TODO: Tell LLM agents what each feature represents.
    #         You can also specify which feature is useful for prediction.
    cols = df.columns.tolist()
    cols_str = ", ".join(cols)
    res = f"The columns are: {cols_str}"

    out.append(res)

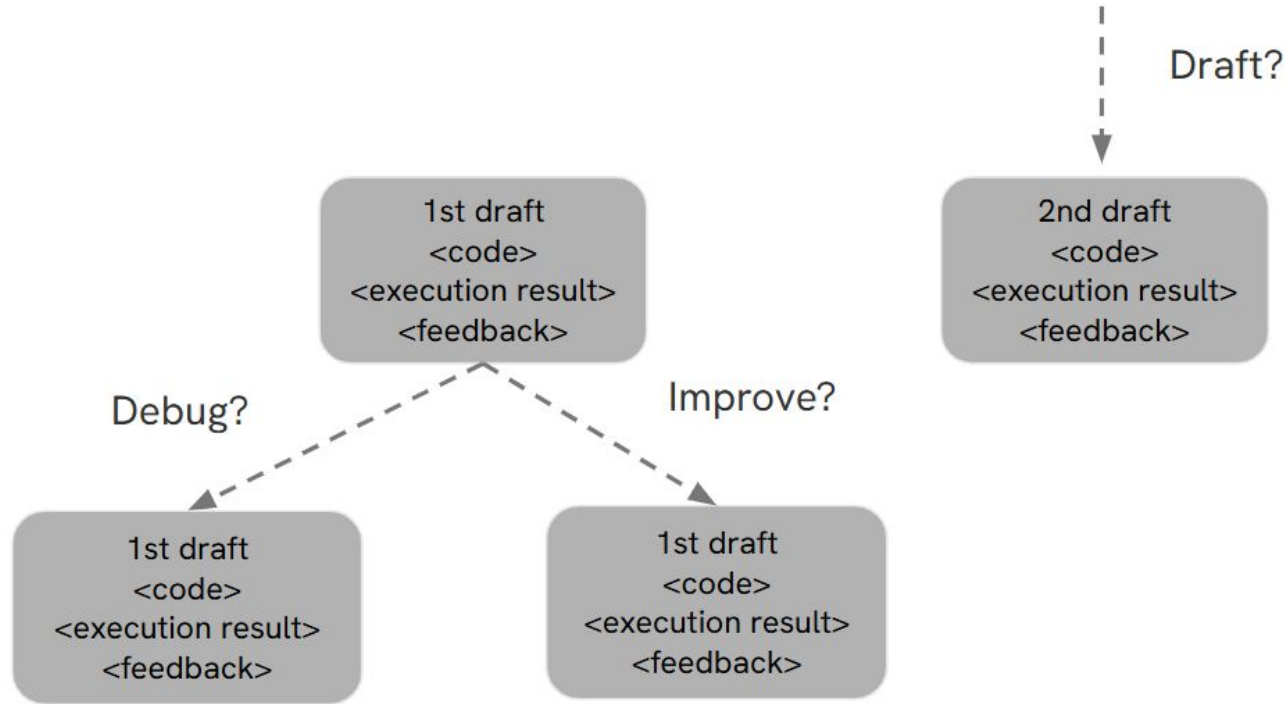
    return "\n".join(out)
```

Hints - Different LLMs

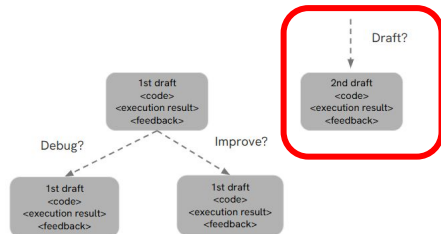
A stronger LLM might be helpful, but not always...

- How could we know which LLM is better?
- Model Size? Pretrained Corpus? [Open LLM Leaderboard](#)?
- You can select the model from [Hugging Face](#).
- Do NOT use closed-source LLM APIs like GPT-4o, Gemini, etc.

Hints - More Iterations



Hints - Drafting More (Optional)



```
def _draft(self) -> Node:

    # ===== TODO: ask LLM agents to come up with a solution and then implement =====

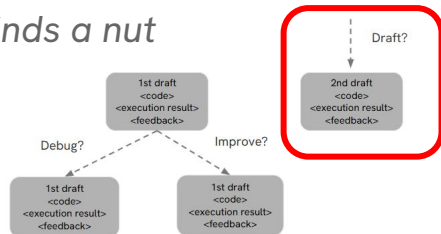
    system_prompt = "You are an AI agent."

    user_prompt = [
        "You have to come up with a solution for machine learning task and then implement this solution in Python.",
        f"The task is to {str(self.cfg.task_goal)} ",
        f'All the provided input data is stored in "{self.cfg.data_dir}" directory.',
        f"{str(self.data_preview)}",
        'You have to save the predictions result on testing set in "/content/submission.csv".',
        'Note that the testing file DOES NOT have the target column.'
    ]

    system_message = system_prompt
    user_message = "¶".join(user_prompt)
    plan, code = self.plan_and_code_query(system_message=system_message, user_message=user_message)
    return Node(plan=plan, code=code)
```

Hints - Drafting More (Optional)

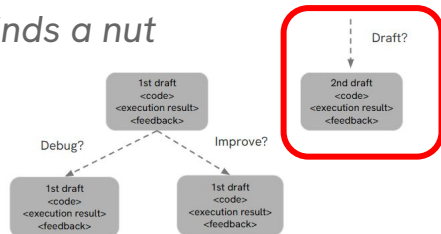
*Even a blind squirrel finds a nut
once in a while*



```
config = {  
    # experiment configurations  
    "exp_name": "ML2025_HW2",  
    "data_dir": Path("/content/ML2025Spring-hw2-public").resolve(),  
  
    # the description of the task  
    "task_goal": "Given the survey results from the past two days in a specific state in the U.S., ¥  
                  predict the probability of testing positive on day 3. ¥  
                  The evaluation metric is Mean Squared Error (MSE).",  
  
    "agent": {  
        # the number of iterations  
        "steps": 1,  
        "search": {  
            # decide whether to debug or improve  
            "debug_prob": 0.5,  
            # the number of draft generated before improving/debugging  
            "num_drafts": 1,  
        },  
    },  
}
```

Hints - Drafting More (Optional)

*Even a blind squirrel finds a nut
once in a while*



```
def set_seed(seed=531):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed(seed)
        torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.benchmark = False
    torch.backends.cudnn.deterministic = True

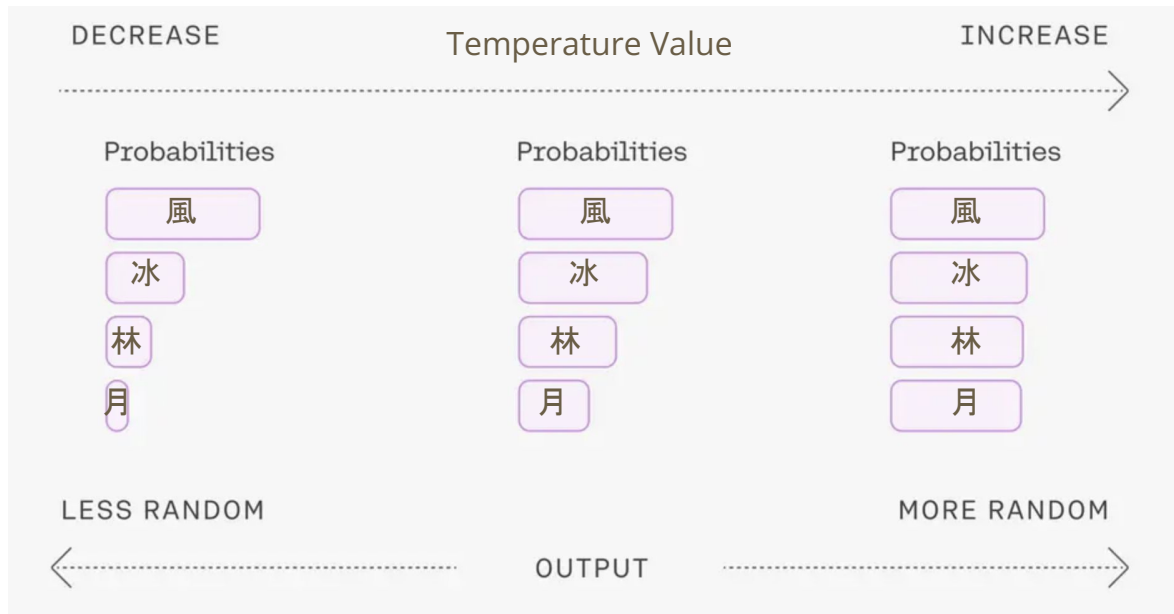
set_seed()
```

```
def generate_response(_model: Llama, _messages: str) -> str:
    """
    This function will inference the model with given messages.
    """
    _output = _model.create_chat_completion(
        _messages,
        stop=["<|eot_id|>", "<|end_of_text|>"],
        max_tokens=4096,      # This argument is how many tokens th
        temperature=0,        # This argument is the randomness of
    )["choices"][0]["message"]["content"]
    return _output
```

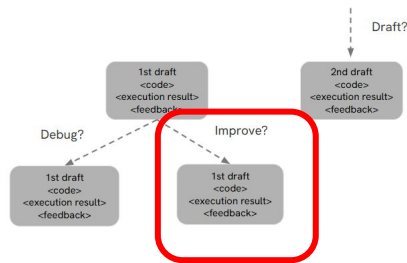
Temperature

This slide is copied from [GenAI2024 HW5](#)

- Related to the diversity of the output, **$0.0 \leq \text{temperature}$**
- Higher temperature for better diversity



Hints - Improving (Optional)



```
def _improve(self, parent_node: Node) -> Node:
```

```
# ===== TODO: ask LLM agent to improve drafts =====
```

```
system_prompt = "You are an AI assistant."
```

```
user_prompt = [
    f"Task description: {str(self.cfg.task_goal)} "
    f"Memory: {str(self.journal.generate_summary())} "
    f"Previous solution: Code: {str(wrap_code(parent_node.code))} "
]
```

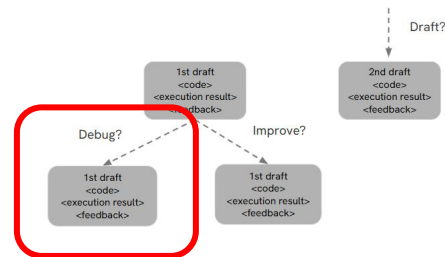
```
system_message = system_prompt
```

```
user_message = " ".join(user_prompt)
```

```
plan, code = self.plan_and_code_query(system_message=system_message, user_message=user_message)
```

```
return Node(plan=plan, code=code, parent=parent_node)
```

Hints - Debugging (Optional)



```
def _debug(self, parent_node: Node) -> Node:

    # TODO: ask LLM agent to debug
    system_prompt = "You are an AI agent."

    user_prompt = [
        f"Task description: {str(self.cfg.task_goal)}\n\n",
        f"Previous (buggy) implementation: {str(wrap_code(parent_node.code))}\n\n",
        f"Execution output: {str(wrap_code(parent_node.term_out, lang=''))}\n\n",
        str(self.data_preview)
    ]

    system_message = system_prompt
    user_message = " ".join(user_prompt)
```


Hints - Evaluating (Optional)

which version of the code to use?

```
def parse_exec_result(self, node: Node, exec_result: ExecutionResult):
    node.absorb_exec_result(exec_result)

    system_prompt = "You are an AI agent."

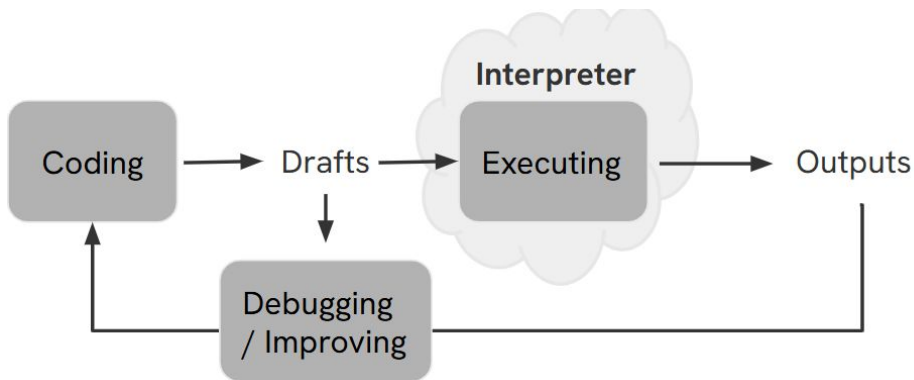
    # TODO: ask LLM agent to extract evaluation result from the
    #       execution output and detect bugs.

    user_prompt = f"""
    The task is:
    {self.task_desc}

    The code implementation is:
    {wrap_code(node.code)}

    The execution output is:
    {wrap_code(node.term_out, lang="")}
    """

    system_message = system_prompt
    user_message = "".join(user_prompt)
```



Structured outputs with llama-cpp-python

<https://python.useinstructor.com/integrations/llama-cpp-python/#patching>

Hints

- When changing LLMs, try to load checkpoints whose size are ~10GBs first
- (For this task) Less is more when it comes to prompt design.
- (For this task) We recommend performing feature selection and using a strong LLM first.
- Feel free to modify the pipeline.

Grading - Regulations

*The LLM agent serves as your representative
—if it violates the rules, it's as if you did.*

- You should NOT plagiarize, if you use any other resource, you should cite it in the reference.
- Do NOT share codes or prediction files with any living creatures.
- Do NOT use any approaches to submit your results more than 5 times a day.
- Do NOT search for or use additional data for training or the answers for the testing data.
- Do NOT use closed-source LLM APIs like GPT-4, Gemini, etc.
- You should NOT modify your input file or prediction files manually.
- Make sure that TAs can reproduce the predictions using the code you submit. (Fix the random seed)
- Your final grade $\times 0.9$ and get a score 0 for that homework if you violate any of the above rules first time (within a semester).
- You will get F for the final grade if you violate any of the above rules multiple (> 1) times.
- Prof. Lee & TAs preserve the rights to change the rules & grades.

Grading - Reproducibility

- If you run your code outside of Colab, please submit it with a README file.
- The format of the README is not restricted, but you must ensure that we can reproduce your results based on your instructions.
- It is better to provide a requirements.txt file.

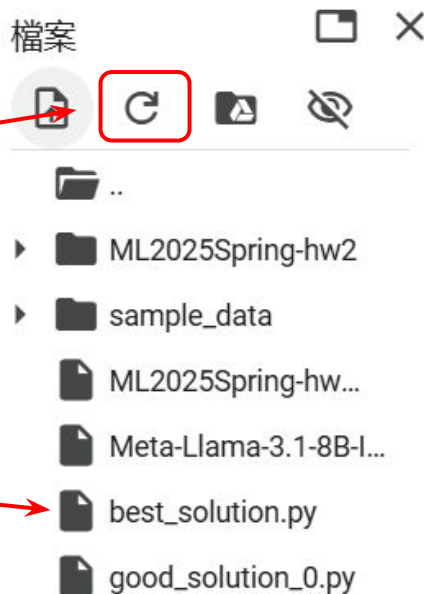
FAQs - No Submission File Generated

Q: I cannot find `submission.csv` after executing code...

A:

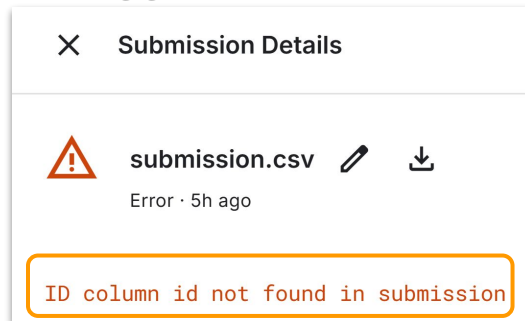
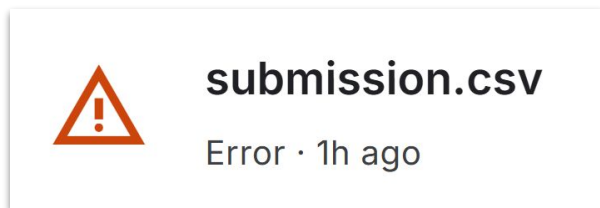
1. Refresh
2. Check `best_solution.py` and adjust your prompts (or try different LLMs)

(Do NOT fix the python script manually).



FAQs - Kaggle Submission Error

Q: When I submit `submission.csv` to Kaggle, an error occurs...

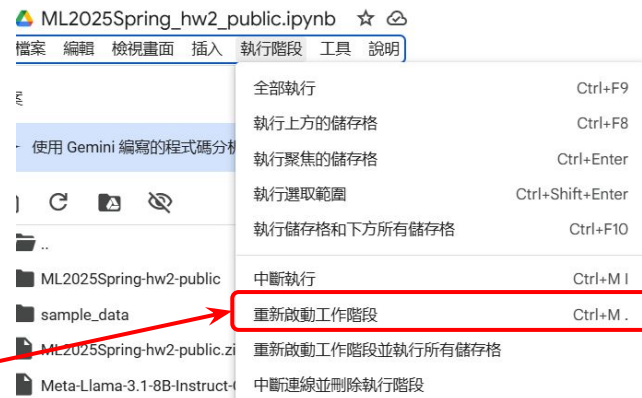


A: Open your `submission.csv`, identify the errors, and adjust the prompts to fix it (Do NOT fix the prediction file manually). Common errors include incorrect column names, the "id" column starting from 1 instead of 0, etc.

FAQs - Error Occurs When Loading LLMs

Q: When I load LLM, an error occurs...

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-5-c1fd42a56efc> in <cell line: 0>()  
      2  
      3 # Load the model onto GPU  
----> 4 myModel = Llama()  
      5 # ===== TODO: try different LLM =====  
      6 # Before changing LLM, restart the session!  
  
-----  
      1 frames  
-----  
/usr/local/lib/python3.11/dist-packages/llama_cpp/_internals.py in __init__(self, model, params, verbose)  
    247  
    248     if ctx is None:  
--> 249         raise ValueError("Failed to create llama_context")  
    250  
    251     self.ctx = ctx  
  
ValueError: Failed to create llama_context
```



A:

1. Try to restart the session.
2. The checkpoint is too large for your GPU. Try a smaller one.

If any questions, you can ask us via...

- ❑ NTU COOL (recommended)

https://cool.ntu.edu.tw/courses/46406/discussion_topics/375678

- ❑ Email

ntu-ml-2025-spring-ta@googlegroups.com

The title should begin with “[HW2]”

- ❑ TA hours

(Fri.) 13:30~14:20

(Fri.) 17:20~18:00