Information Security [Progress]

19102096 Donghwan Lee
19102127 Suho Lee

TABLE OF CONTENTS

01

Remind

What we’ve done?

02

Progress

Data Preprocessing & Code, Execution Explanation

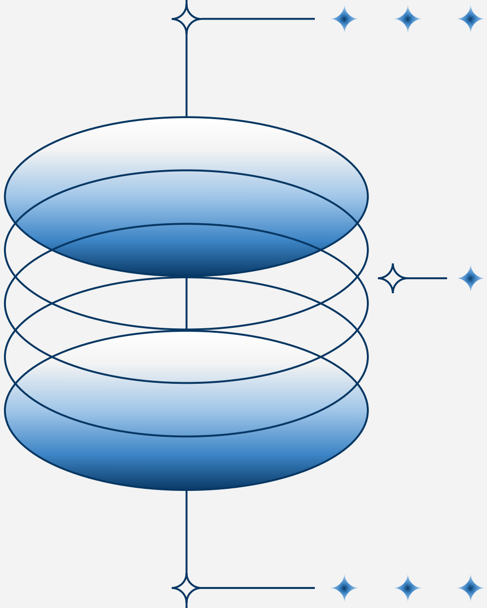
03

Plan

What We’re going to do?

01

Remind



Remind the Origin Plan & Status

Secure credit scoring system

Overview

- A person's credit information is stored in homomorphic encryption, and a bank or financial institution can perform to calculate a credit score on this encrypted data.

- This way, your credit information is not exposed to the outside world, but the financial institution can still make a credit assessment based on the information they need.

Our Credit Score Formula

How do We set Credit Score Formula?

- This score can be used by lenders to determine whether to issue you a credit card or not, or whether you're a good or bad credit risk.
- However, the formula for calculating credit scores from “KCB” and “Nice” is not publicly available, so we don't know the formula.
- Therefore...

Replaced with a credit score calculation formula ***we built ourselves***

Credit scoring methods & criteria

How do we set Credit Scoring Formula?

- A financial institution wants to build predictive and classification models using direct homomorphic encrypted data to determine customers instead of inaccurate credit scores.
- User 16 variables

Credit Score =(w1xGender)+(w2×Whether you have a car)+(w3×Whether you have a own property)+(w4×Whether you own phone for workplace)+(w5×Whether you own phone for dailylife)+(w6×whether you have your own Email)+(w7×whether you have own job)+(w8×number of children you have)+(w9×Length of transaction period)+(w10×Amount of total income)+(w11×Your age)+(w12×Income type)+(w13×Educational type)+(w14×Family status)+(w15×housing type)+(w16xOccupation type)

Credit scoring methods & criteria

How do we get weight(w)?

1. Feature importance

⇒Since We result in a numerical value in decimal form, combine them to calculate the weight

And Finally…

We can calculate the multiplication of feature importance and user input variables
Then get a Financial Score

02

Progress

Interim reporting of implementation status,
How to execute implementation, How to check results

Current Implementation Status

Status

1. Data for project is now completely preprocessed.

2. Can get user data(18 variables) from User.

3. User Input data Encryption is completed

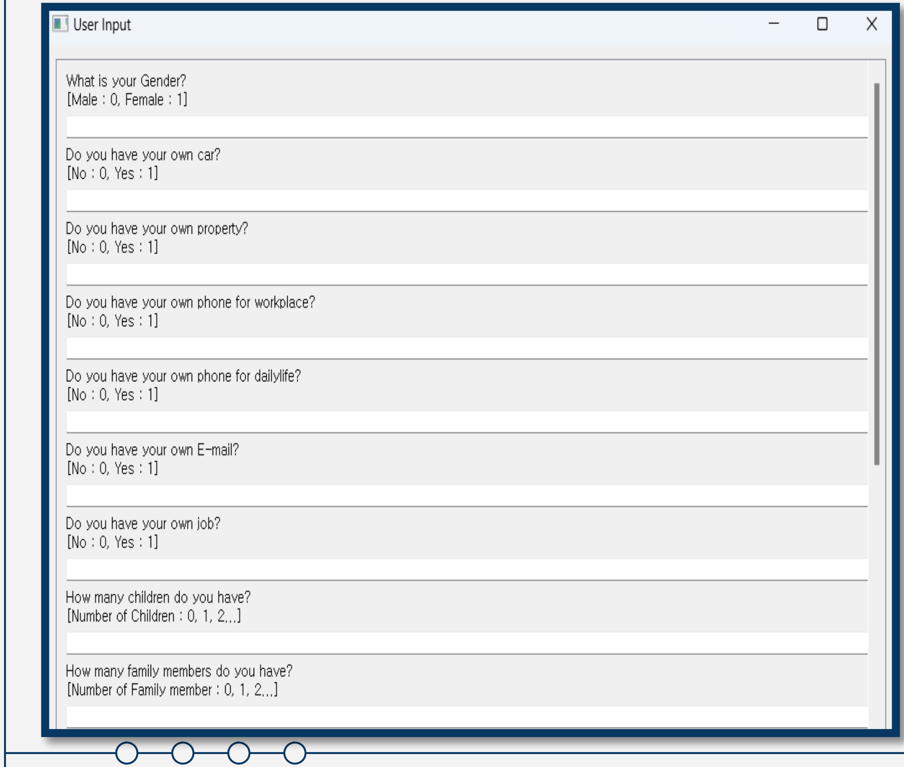
4. Get its own Feature_importance from our preprocessed data and can Encrypt its value.

5. Calculate the Financial Score with Pi -Heaan and Decrypt it for Financial Score .

6. Prepare the plaintext of Decision Tree and Random forest for the final

=> So We can Finally get a User Financial data from our own Formula.

How to execute the implementation



What is your Gender?
[Male : 0, Female : 1]

Do you have your own car?
[No : 0, Yes : 1]

Do you have your own property?
[No : 0, Yes : 1]

Do you have your own phone for workplace?
[No : 0, Yes : 1]

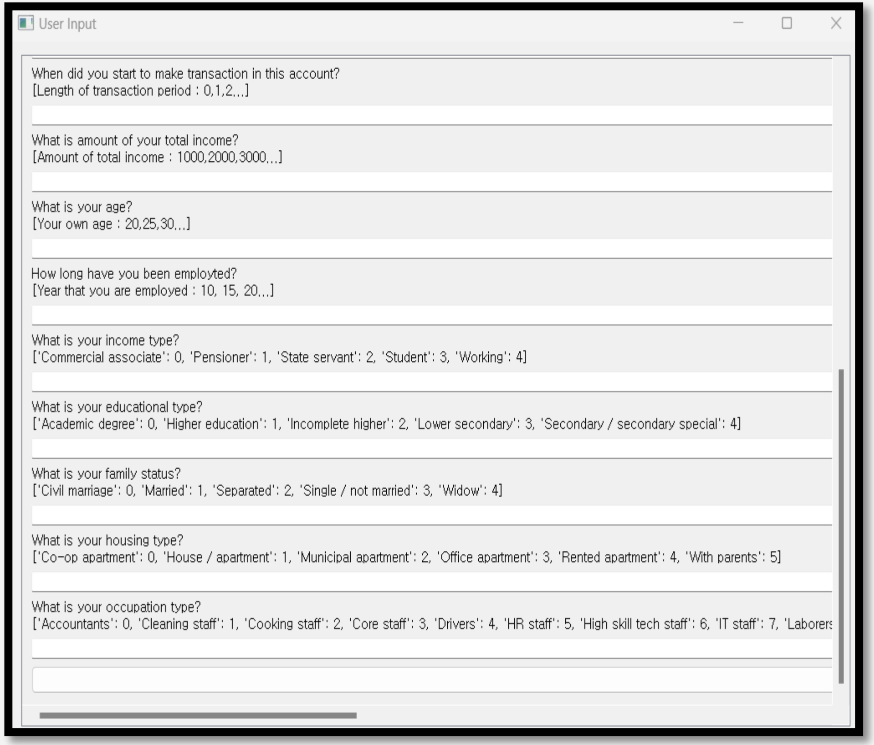
Do you have your own phone for dailylife?
[No : 0, Yes : 1]

Do you have your own E-mail?
[No : 0, Yes : 1]

Do you have your own job?
[No : 0, Yes : 1]

How many children do you have?
[Number of Children : 0, 1, 2,..]

How many family members do you have?
[Number of Family member : 0, 1, 2,..]



When did you start to make transaction in this account?
[Length of transaction period : 0,1,2,..]

What is amount of your total income?
[Amount of total income : 1000,2000,3000,..]

What is your age?
[Your own age : 20,25,30,..]

How long have you been employted?
[Year that you are employed : 10, 15, 20,..]

What is your income type?
['Commercial associate': 0, 'Pensioner': 1, 'State servant': 2, 'Student': 3, 'Working': 4]

What is your educational type?
['Academic degree': 0, 'Higher education': 1, 'Incomplete higher': 2, 'Lower secondary': 3, 'Secondary / secondary special': 4]

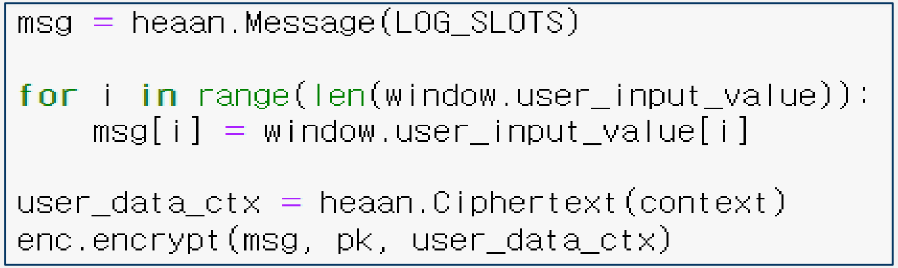
What is your family status?
['Civil marriage': 0, 'Married': 1, 'Separated': 2, 'Single / not married': 3, 'Widow': 4]

What is your housing type?
['Co-op apartment': 0, 'House / apartment': 1, 'Municipal apartment': 2, 'Office apartment': 3, 'Rented apartment': 4, 'With parents': 5]

What is your occupation type?
['Accountants': 0, 'Cleaning staff': 1, 'Cooking staff': 2, 'Core staff': 3, 'Drivers': 4, 'HR staff': 5, 'High skill tech staff': 6, 'IT staff': 7, 'Laborers

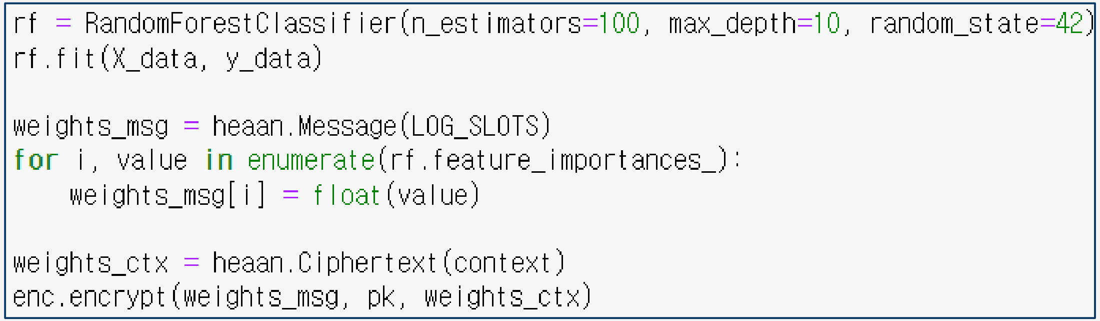
=> User input 16 values through input command

How to execute the implementation



```
msg = heaan.Message(LOG_SLOTS)

for i in range(len(window.user_input_value)):
    msg[i] = window.user_input_value[i]

user_data_ctx = heaan.Ciphertext(context)
enc.encrypt(msg, pk, user_data_ctx)
```

- Encrypts user -entered data and stores it in a vector in user_data_ctx



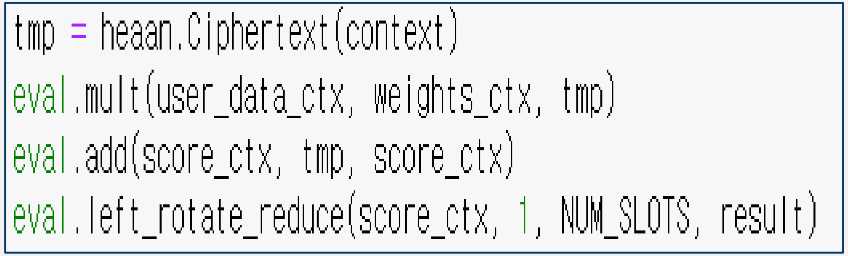
```
rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
rf.fit(X_data, y_data)

weights_msg = heaan.Message(LOG_SLOTS)
for i, value in enumerate(rf.feature_importances_):
    weights_msg[i] = float(value)

weights_ctx = heaan.Ciphertext(context)
enc.encrypt(weights_msg, pk, weights_ctx)
```

- Learn RF models with plaintext data and obtain feature_importance

- Encrypt the Weighted value

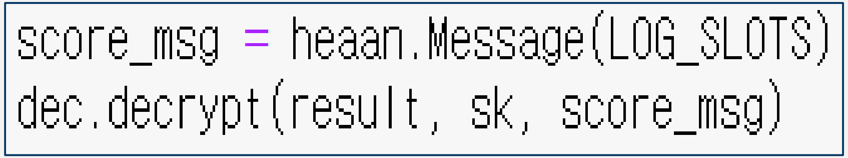
How to check the results

Calculation



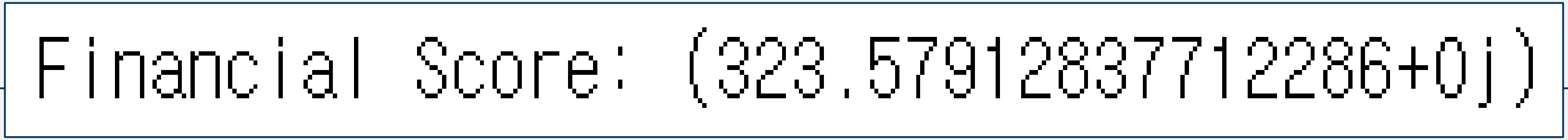
```
tmp = heaan.Ciphertext(context)
eval.mult(user_data_ctx, weights_ctx, tmp)
eval.add(score_ctx, tmp, score_ctx)
eval.left_rotate_reduce(score_ctx, 1, NUM_SLOTS, result)
```

- Multiply the first attribute by the weight and store it in score_ctx
- Multiply the remaining attributes and weights, and add them to the score_ctx



```
score_msg = heaan.Message(LOG_SLOTS)
dec.decrypt(result, sk, score_msg)
```

- Decrypts stored financial scores as a result of the operation



```
Financial Score: (323.57912837712286+0j)
```

03

What We're going to do?

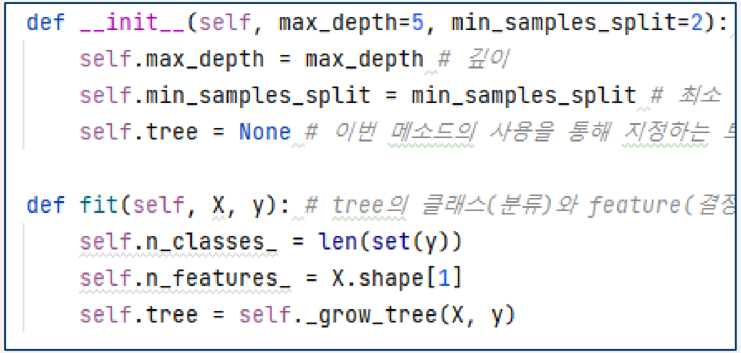
[Finance] Secure credit scoring system

Future Implementation

Implemented & plan

- Currently, we are using the ' sklearn ' library - 'feaure_importance ' method from random forest to **get the weight for each score**
- However, knowing the formula for calculating the weights is not appropriate for the security that **this project is aiming for**
- To compensate for this, we are planning to implement the part that uses the library related to random forest directly in plaintext code to fully encrypt the entire process
- For now, we have written Python code to implement parts about the random forest and the decision trees that comprise it **as a previous step**

Plaintext code – Decision Tree

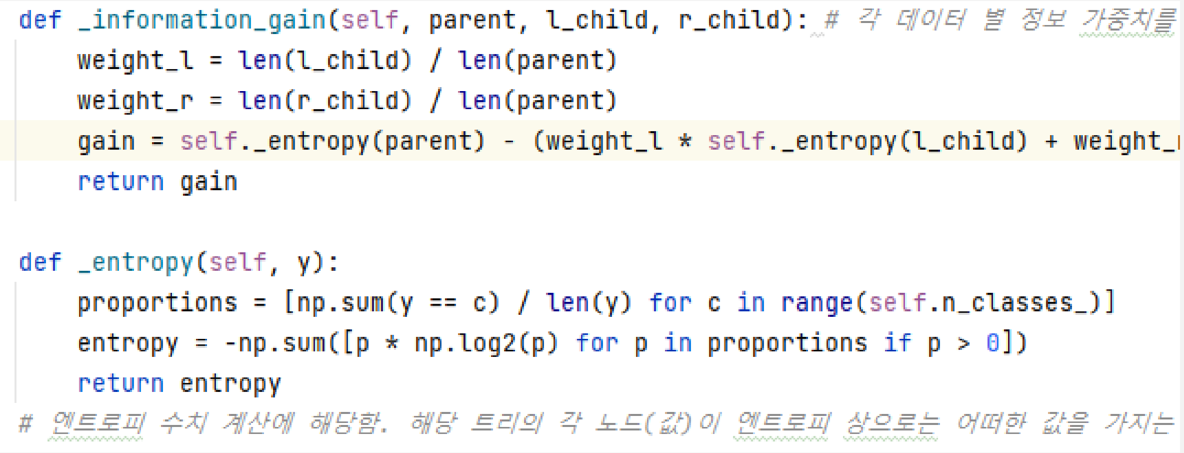


```
def __init__(self, max_depth=5, min_samples_split=2):
    self.max_depth = max_depth # 깊이
    self.min_samples_split = min_samples_split # 최소
    self.tree = None # 이번 메소드의 사용을 통해 지정하는

def fit(self, X, y): # tree의 클래스(분류)와 feature(결
    self.n_classes_ = len(set(y))
    self.n_features_ = X.shape[1]
    self.tree = self._grow_tree(X, y)
```

- Set the criteria for generating the tree as a parameter like we would normally set in a library

- Fit each property in the tree to the size of the dataset given as input values

Plaintext code – Decision Tree

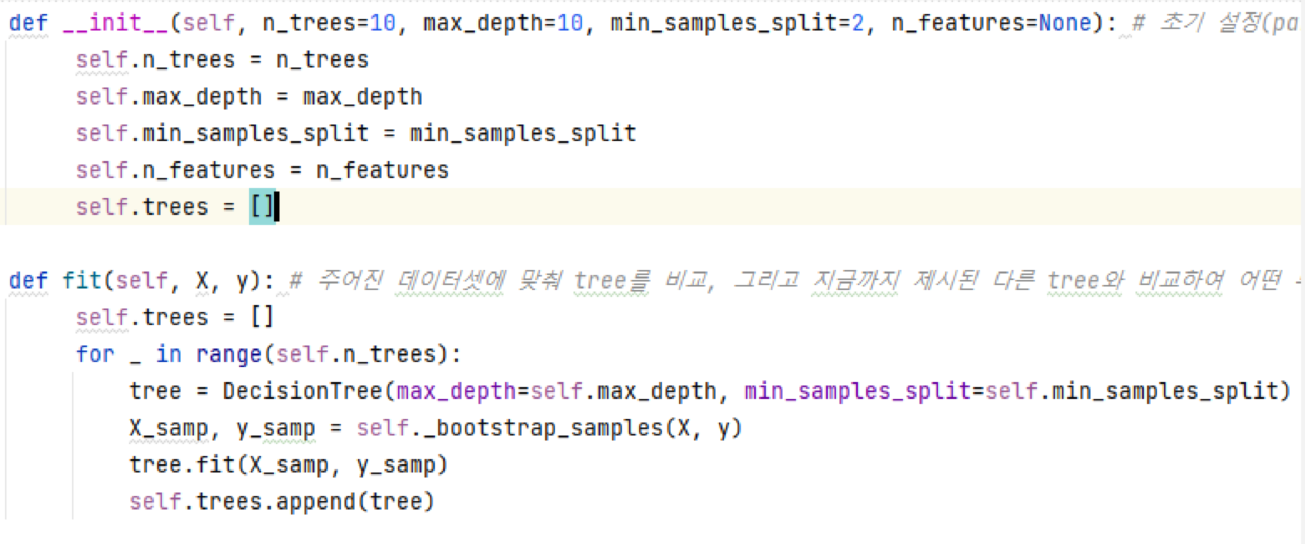


```
def _information_gain(self, parent, l_child, r_child): # 각 데이터 별 정보 가중치를
    weight_l = len(l_child) / len(parent)
    weight_r = len(r_child) / len(parent)
    gain = self._entropy(parent) - (weight_l * self._entropy(l_child) + weight_
    return gain

def _entropy(self, y):
    proportions = [np.sum(y == c) / len(y) for c in range(self.n_classes_)]
    entropy = -np.sum([p * np.log2(p) for p in proportions if p > 0])
    return entropy
# 엔트로피 수치 계산에 해당함. 해당 트리의 각 노드(값)이 엔트로피 상으로는 어떠한 값을 가지는
```

Classify a tree by measuring the **information gain** for each class

with **entropy** as the tree classification criterion

Plaintext code – Random Forest

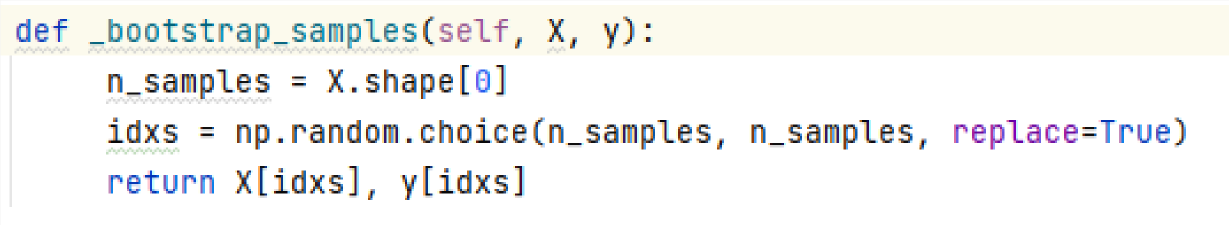


```
def __init__(self, n_trees=10, max_depth=10, min_samples_split=2, n_features=None): # 초기 설정(pa
    self.n_trees = n_trees
    self.max_depth = max_depth
    self.min_samples_split = min_samples_split
    self.n_features = n_features
    self.trees = []

def fit(self, X, y): # 주어진 데이터셋에 맞춰 tree를 비교, 그리고 지금까지 제시된 다른 tree와 비교하여 어떤 
    self.trees = []
    for _ in range(self.n_trees):
        tree = DecisionTree(max_depth=self.max_depth, min_samples_split=self.min_samples_split)
        X_samp, y_samp = self._bootstrap_samples(X, y)
        tree.fit(X_samp, y_samp)
        self.trees.append(tree)
```

Since a random forest is about **choosing the best one** among multiple decision trees,

it follows from the decision tree code we defined **previously**

Plaintext code – Random Forest

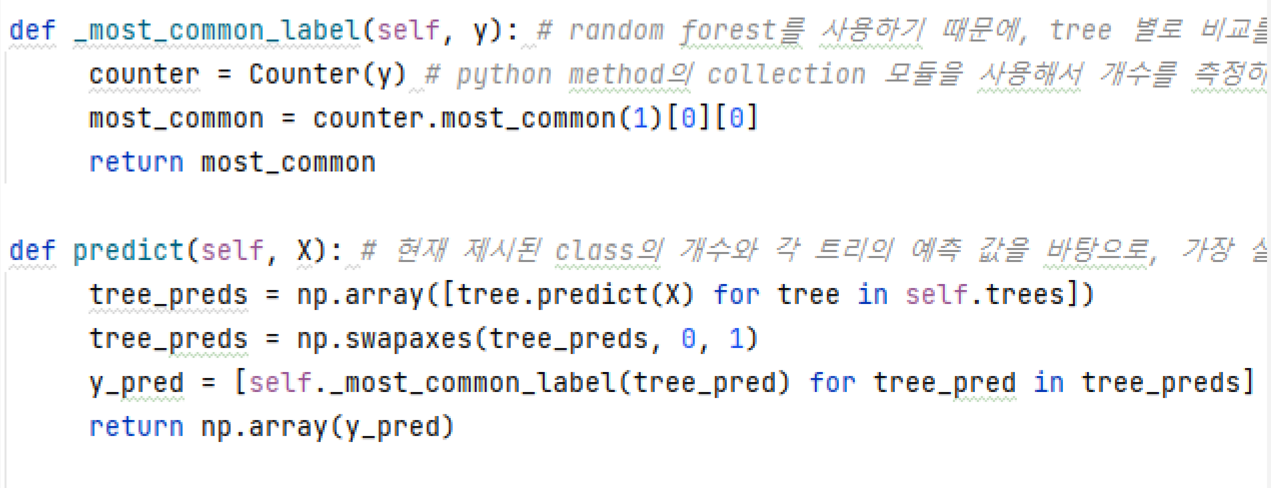


```python
def _bootstrap_samples(self, X, y):
    n_samples = X.shape[0]
    idxs = np.random.choice(n_samples, n_samples, replace=True)
    return X[idxs], y[idxs]
```

Determine whether the tree trained on

this round has any **performance advantage** over the others

Plaintext code – Random Forest



```python
def _most_common_label(self, y): # random forest를 사용하기 때문에, tree 별로 비교를
    counter = Counter(y) # python method의 collection 모듈을 사용해서 개수를 측정하
    most_common = counter.most_common(1)[0][0]
    return most_common

def predict(self, X): # 현재 제시된 class의 개수와 각 트리의 예측 값을 바탕으로, 가장 실
    tree_preds = np.array([tree.predict(X) for tree in self.trees])
    tree_preds = np.swapaxes(tree_preds, 0, 1)
    y_pred = [self._most_common_label(tree_pred) for tree_pred in tree_preds]
    return np.array(y_pred)
```

Classify the tree models previously trained on the data into each class

to measure which of those trees is **best suited to classify** the current dataset

Future Plan

Next & Finalgoal

1. **Change** the currently presented plaintext code to a form suitable for **homomorphic encryption operations**

2. **Analyze** the contents of the ' Feature_importance ' library and write , homomorphically encrypt it in plain text

3. Decide whether to issue a card to a user as good or bad based on the financial score output **as a result of the calculation**

4. **Test** the actual issuance of the card , using a real randomized test dataset or example users

5. Present the **final results**

OUR TEAM

Donghwan Lee

ITM 19102096

Suho Lee

ITM 19102127