

Sec 26 กลุ่มที่ 7

รายงานเรื่อง AI แนะนำและจำแนกชีส

โดย

65010188 ชยพล ลำเทียน

65010319 ณัฐภัทร เอกชน

65010321 ณัฐรัตน์ หวังใจดี

65010356 ดิฉัน แยมพันธ์

65010556 บวรพจน์ พวงทอง

65010659 พงศ์พล วิวัฒน์สันติวงศ์

1. Project detail (หัวข้อเดียวกัน)

สมัยนี้ต้องยอมรับเลยว่า “ชีส” เป็นอาหารและวัตถุดิบหลักชนิดหนึ่งของอาหารตะวันตก เป็นอาหารที่มีการบริโภคมากติดอันดับต้นๆของโลก โดยปัจจุบันมีประเภทชีสมากกว่า 3,000 ชนิดทั่วโลก ที่มีความแตกต่างกันทั้งในด้านประเภทนม เนื้อสัมผัส รสชาติ กลิ่น สี รวมถึงชีสสำหรับมังสวิรัต และ Vegan ที่ต้องเฉพาะเจาะจงเข้าไปอีก นี่ยังไม่รวมถึงประเทศในการผลิตชีส ที่ทำให้มีวัตถุดิบในการทำที่เหมือนกัน แต่อยู่ต่างสถานที่ ก็ถือว่าเป็นคนละชนิดกัน ดังนั้นการที่จะเลือกชีสสักตัวมาใช้ประกอบอาหาร หรือจับคู่กับเครื่องดื่มนั้น ถือเป็นเรื่องที่ค่อนข้างท้าทาย

ในปัจจุบันมีเว็บไซต์ต่างประเทศที่เกี่ยวกับชีสอยู่มากมาย ยกตัวอย่างเช่น Cheese.com เป็นเว็บไซต์ที่เก็บข้อมูลเกี่ยวกับประเภทของชีส แหล่งที่มา และรสชาติ หรือ The Cheese Professor เป็นเว็บไซต์ที่เน้นการจับคู่ชีสกับอาหารและเครื่องดื่มที่หลากหลาย และยังมีอีกหลากหลายเว็บไซต์ เช่น Murray's Cheese หรือ Speciality Food Magazine ที่เน้นไปที่ผลิตภัณฑ์ของชีส รวบรวมเมนูอาหารต่างๆที่มีชีสเข้ามาเกี่ยวข้อง หรือการขายชีสออนไลน์

จากที่กล่าวข้างต้นเว็บไซต์ที่มีอยู่ในตอนนี้ มักมีข้อมูลชีสที่แบ่งตามประเภทหรือให้คำแนะนำทั่วไปเกี่ยวกับการจับคู่อาหาร ซึ่งผู้ใช้สามารถเข้าไปเรียนรู้หรือหาข้อมูลได้ตามหมวดหมู่ที่ที่นักพัฒนาได้ทำการกำหนดไว้ บางเว็บไซต์ที่ขายชีสออนไลน์ยังช่วยให้สามารถเข้าถึงชีสจากทั่วโลกได้ง่ายขึ้น รวมถึงบางเว็บไซต์ยังสามารถแลกเปลี่ยนความคิดเห็น หรือแชร์ประสบการณ์เกี่ยวกับชีส ซึ่งช่วยให้สร้างชุมชนสำหรับคนรักชีสขึ้นมาได้ด้วย แต่ถึงอย่างนั้นเว็บไซต์ส่วนใหญ่ หรือเกือบทั้งหมดไม่ได้ให้การแนะนำที่เฉพาะเจาะจงตามความต้องการของผู้ใช้ ทำให้ไม่สามารถเข้าถึงคำแนะนำที่ตรงตามรสนิยมของแต่ละคนได้

ทำให้เราเล็งเห็นว่าในเรื่องเว็บไซต์ของชีส ยังไม่มีเว็บไซต์ไหนในต่างประเทศ โดยเฉพาะในไทยที่นำ AI เข้ามาใช้ในการเลือกหรือแนะนำชีส ซึ่งสามารถทำให้ผู้ใช้ระบุลักษณะของชีสที่ต้องการได้ ได้แก่ ประเภทนม เนื้อสัมผัส รสชาติ กลิ่น สี รวมถึงชีสสำหรับมังสวิรัต และ Vegan หลังจากนั้นจะใช้ AI เข้ามาช่วยในการเลือกชนิดของชีสที่เหมาะสมที่สุดสำหรับผู้ใช้ รวมถึงแนะนำการจับคู่ชีสกับอาหารและเครื่องดื่ม รวมถึงรายละเอียดอื่นๆอีกด้วย

ดังนั้นเราจึงคิดว่าเว็บไซต์ “AI แนะนำและจำแนกชีส” จะสามารถเข้ามาแก้ปัญหาการเลือกชีสในการนำไปเป็นอาหารและวัตถุดิบ รวมถึงจับคู่กับเครื่องดื่มได้อย่างมีประสิทธิภาพ ซึ่งในอนาคตจะสามารถพัฒนาต่อไปได้จนเข้าไปเป็นส่วนช่วยสำคัญทั้งในด้านครัวเรือน และอุตสาหกรรมอาหาร

2. วิธี implement อย่างละเอียด

2.1 ในส่วนของ การ Cleaning Data แล้วนำไป Train Model

ในขั้นตอนแรกเราจะทำการ Clean Data ที่เรา Import ไฟล์.csv เข้ามา

```
In [2]: df = pd.read_csv("balanced_cheese_family2.csv")
```

โดยเราจะครีโอล column ที่ไม่จำเป็นในการเทรนก่อน

```
In [3]: df.drop('url',axis=1,inplace=True)
df.drop('region',axis=1,inplace=True)
df.drop('synonyms',axis=1,inplace=True)
df.drop('alt_spellings',axis=1,inplace=True)
df.drop('producers',axis=1,inplace=True)
df.drop('calcium_content',axis=1,inplace=True)
```

หลังจากนั้นจะแปลงค่า % ใน column fat_content ให้เป็นช่วง High/Medium/Low โดยอิงจากค่า Max และ Min

```
In [6]: # เปลี่ยน fat_content จาก % เป็นช่วง LOW MEDIUM HIGH
# แปลงค่าเปอร์เซ็นต์เป็นตัวเลห
percentages = df['fat_content'].unique()
percentages = [float(p.strip('%')) for p in percentages]

# คำนวณค่าสูงสุดและต่ำสุด
min_value = min(percentages)
max_value = max(percentages)

# คำนวณช่วงสำหรับ High, Medium, Low
low_threshold = min_value + (max_value - min_value) / 3
medium_threshold = min_value + 2 * (max_value - min_value) / 3

In [7]: # แปลงค่าเปอร์เซ็นต์ในคอลัมน์เป็นตัวเลห
df['fat_content'] = df['fat_content'].str.rstrip('%').astype(float)

# ฟังก์ชันในการแปลงค่าเป็นช่วง Low, Medium, High
def categorize_fat_content(value):
    if value <= low_threshold:
        return 'Low'
    elif value <= medium_threshold:
        return 'Medium'
    else:
        return 'High'

# ใช้ฟังก์ชันเพื่อแปลงค่าคอลัมน์ 'fat_content'
df['fat_content'] = df['fat_content'].apply(categorize_fat_content)
```

แล้วทำการเช็คและลบ row ที่มีค่า Nan มากกว่าค่า จากนั้นจะทำการแบ่งสัดส่วนของข้อมูลแต่ละค่า แล้วทำการเติมค่า Nan ด้วยข้อมูลที่มีจำนวนน้อยก่อน เพื่อให้ค่าเฉลี่ยของข้อมูลใกล้เคียงกัน

```
In [8]: # นับจำนวนค่าว่างในแต่ละแถว
nan_count = df.isna().sum(axis=1)
df = df.drop(index=df[nan_count > 5].index)
df.info()
```

```
In [9]: # ทำการเติมแบบแบ่งสัดส่วน most_frequent
# แบ่งสัดส่วนในการเติมค่า NaN

# ทำการเติมแบบแบ่งสัดส่วน most_frequent
for col in df.select_dtypes("object").columns:
    # คำนวณการแจกแจงของค่าที่มีในคอลัมน์
    value_counts = df[col].value_counts(normalize=True)

    # สุ่มค่าเพื่อเติม NaN ตามสัดส่วนที่ได้
    imputed_values = np.random.choice(value_counts.index,
                                       size=df[col].isna().sum(),
                                       p=value_counts.values)

    # เติมค่า NaN ใน DataFrame
    df.loc[df[col].isna(), col] = imputed_values
```

และในขั้นตอนสุดท้ายของส่วนนี้คือการ Export ไฟล์ออก

```
In [11]: df.to_csv('cheese_before_encode_5.csv', index=False)
```

ในส่วนถัดไปจะเป็นในส่วนการ Encode ข้อมูลทุก column ให้เป็น str

```
In [12]: from sklearn import preprocessing
# สร้าง dictionary สำหรับการเก็บ mapping ที่แน่นอน
static_mapping = {}

# แปลงทุกคอลัมน์ใน DataFrame ให้เป็น string
for col in df.columns:
    # แปลงคอลัมน์ boolean เป็น string และคอลัมน์อื่น ๆ เป็น string ด้วย
    df[col] = df[col].astype(str)

    # ถ้ายังไม่มี mapping สำหรับคอลัมน์นี้ ให้สร้างใหม่
    if col not in static_mapping:
        le = preprocessing.LabelEncoder()
        le.fit(df[col])
        static_mapping[col] = dict(zip(le.classes_, le.transform(le.classes_)))

# ใช้ static_mapping ในการเข้ารหัส DataFrame
for col, map_dict in static_mapping.items():
    df[col] = df[col].map(map_dict)

# แสดงผลลัพธ์ DataFrame หลังจากการแปลง
print(static_mapping)
print(df)

# แสดงว่าข้อมูลแต่ละ column map กับเลขไหน
for col, map_dict in static_mapping.items():
    print(f"Static Mapping for {col}: {map_dict}")
```

แล้วทำการ Export ไฟล์ออก

```
In [15]: df.to_csv('cheese_encode_5.csv', index=False)
```

2.2 กระบวนการ Train และ Test Model AI

โดยในส่วนนี้เป็นการทำ classification โดยก่อนหน้านี้นี้ได้ลองหลายโมเดลเช่น Decision Tree , Naive Bayes , KNN และ Random Forest โดยได้ลองทุกโมเดล จะพบว่าโมเดล Random Forest มีประสิทธิภาพมากที่สุด สำหรับ Dataset นี้ โดยผมได้ลอง import model Random forest ของ sklearn มีประสิทธิภาพดีพอสมควร accuracy อยู่ที่ประมาณ 80% แต่ผมต้องการเข้าใจการทำงานของ model นี้และสามารถควบคุม Parameter ต่างๆ ได้ จึงได้ทำการสร้าง model Random forest ขึ้นมาเอง โดยหลักการก็คือการสร้างจาก Decision Tree classifier เป็นฐานสำหรับการสร้าง decision tree หลายๆต้น เพื่อรวมผลการคาดการณ์จาก decision tree หลายๆต้น โดยกำหนดให้ test set เป็น 30% ของ dataset และ train set 70% โดยกำหนดจำนวนต้นไม้(n_estimator) เท่ากับ 100 ความลึกสูงสุดของต้นไม้(max_depth) เท่ากับ 12 และกำหนด sample ขั้นต่ำที่ใช้ในการแยกข้อมูล 3 ตัวอย่าง แล้วนำ Train set, Test set, และค่า Hyperparameter ไปทำการ train model ออกมาจะได้ค่าความแม่นยำอยู่ที่ 91%

(เปิดดู Code ได้ที่ https://drive.google.com/drive/folders/1RS78-nEY2SqvKA73OdfO0Uk-5dtnWxCf?fbclid=IwY2xjawF5pXpleHRuA2FlbQlXMAABHf4zSamcpouGuBe0WB3xwFh5kslOLb_ZynpfrwFr9MqXwyld80lwbNp9Q_aem_bVMTwvcloMLzU4Z5KQSeMg)

ทำการดึง dataset

```
In [ ]: data = pd.read_csv("cheese_encode_5.csv")
data.head()
```

Out[3]:

	cheese	milk	country	family	type	fat_content	texture	rind	color	flavor	aroma	vegetarian	vegan
0	2	4	72	3	47	0	11	10	16	596	49	0	0
1	3	18	35	3	39	0	114	7	16	69	226	1	0
2	9	4	31	3	38	0	99	7	11	470	0	1	0
3	11	4	35	3	39	0	93	7	11	488	57	0	0
4	15	18	74	3	21	0	109	7	15	71	158	1	0

Node Class

Node Class เป็นส่วนสำคัญในการเก็บข้อมูลสำหรับแต่ละ "โหนด" (Node) ของต้นไม้ ซึ่งแต่ละโหนดจะเป็นจุดที่เราทำการตัดสินใจว่าจะไปในทิศทางไหนต่อจากข้อมูลที่มี โดยใช้ฟิเจอร์และ threshold ในการกำหนด

```
In [ ]: # Decision Tree Node Class (Unchanged)
class Node():
    def __init__(self, feature_index=None, threshold=None, left=None, right=None, info_gain=None, value=None):
        self.feature_index = feature_index # กำหนดอินเด็กซ์ของฟิเจอร์ที่ใช้ในการแบ่งข้อมูล
        self.threshold = threshold # กำหนดค่าขีดจำกัดในการแบ่งข้อมูลของฟิเจอร์นั้น
        self.left = left # โหนดซ้าย (subtree ซ้าย)
        self.right = right # โหนดขวา (subtree ขวา)
        self.info_gain = info_gain # ค่าข้อมูลที่ได้รับการแบ่ง (information gain)

        self.value = value # ค่าของโหนด (ใช้ใน Leaf nodes)
```

Decision Tree Classifier

ทำหน้าที่ในการสร้างและใช้งาน Decision Tree

ภาพรวมการทำงานมี 3 ส่วน ที่สำคัญดังนี้

1. สร้างต้นไม้: ฟังก์ชัน build_tree จะสร้างต้นไม้ตัดสินใจจากข้อมูลโดยใช้กระบวนการค้นหาการแบ่งที่ดีที่สุดในแต่ละโหนด
2. การแบ่งข้อมูล: ฟังก์ชัน get_best_split ค้นหาการแบ่งที่เหมาะสมที่สุดโดยเลือกฟิเจอร์และค่าเกณฑ์ที่เ็นค่า Information Gain สูงสุด
3. ทำนายค่า: ฟังก์ชัน predict และ make_prediction ทำหน้าที่ทำนายค่าจากข้อมูลใหม่ โดยใช้โหนดต่าง ๆ ที่ถูกสร้างขึ้นจากต้นไม้

```
In [ ]: # Decision Tree Classifier (Unchanged)
class DecisionTreeClassifier():
    # ฟังก์ชันนี้เป็นการกำหนดค่าพื้นฐานของคลาส เช่น จำนวนตัวอย่างข้อมูลขั้นต่ำที่ใช้ในการแบ่ง
    # (min_samples_split) และความลึกสูงสุดของต้นไม้ (max_depth)
    def __init__(self, min_samples_split=2, max_depth=2):
        self.root = None
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth

    # เป็นฟังก์ชันที่สร้างต้นไม้โดยใช้การแบ่งข้อมูลแบบ recursive(ซ้ำ)
    def build_tree(self, dataset, curr_depth=0):
        X, Y = dataset[:, :-1], dataset[:, -1] # แยกข้อมูลออกเป็นฟีเจอร์ (X) และเป้าหมาย (Y)
        num_samples, num_features = np.shape(X)

        if num_samples >= self.min_samples_split and curr_depth <= self.max_depth:
            # ค้นหาเกณฑ์ที่ดีที่สุด
            best_split = self.get_best_split(dataset, num_samples, num_features)

            # เมื่อได้ค่าที่ดีที่สุดแล้วจะสร้างโหนดซ้าย-ขวาจนถึงความลึกสูงสุดของ info_gain
            if best_split and best_split["info_gain"] > 0:
                left_subtree = self.build_tree(best_split["dataset_left"], curr_depth + 1)
                right_subtree = self.build_tree(best_split["dataset_right"], curr_depth + 1)
                return Node(best_split["feature_index"], best_split["threshold"],
                            left_subtree, right_subtree, best_split["info_gain"])

            # กรณีไม่สามารถแบ่งได้แล้ว(best_split < 0) จะคำนวณค่าที่ดีที่สุดของ Leaf node
            leaf_value = self.calculate_leaf_value(Y)
            return Node(value=leaf_value)

    # หากการแบ่งที่ดีที่สุดโดยใช้การคำนวณค่าข้อมูลที่ได้รับ (information gain) ซึ่งขึ้นอยู่กับเกณฑ์ Gini index หรือ entropy
    def get_best_split(self, dataset, num_samples, num_features):
        # เตรียมการเก็บค่าการแบ่งที่ดีที่สุด
        best_split = {}
        max_info_gain = -float("inf")

        # วนลูปผ่านทุกฟิเจอร์ในข้อมูล เพื่อทำการตรวจสอบว่าการแบ่งตามฟิเจอร์นั้นๆ จะให้ผลลัพธ์ที่ดีหรือไม่
        for feature_index in range(num_features):
            feature_values = dataset[:, feature_index]
            possible_thresholds = np.unique(feature_values)

            # วนลูปผ่านค่าขีดจำกัดที่เป็นไปได้ ผ่านแต่ละ threshold ที่เป็นไปได้สำหรับฟิเจอร์นั้นๆ
            for threshold in possible_thresholds:
                # dataset_left: ข้อมูลที่มีค่าน้อยกว่าหรือเท่ากับ threshold
                # dataset_right: ข้อมูลที่มีค่ามากกว่า threshold
                dataset_left, dataset_right = self.split(dataset, feature_index, threshold)

                # ตรวจสอบว่าการแบ่งมีข้อมูลทั้งสองด้านหรือไม่
                if len(dataset_left) > 0 and len(dataset_right) > 0:
                    y, left_y, right_y = dataset[:, -1], dataset_left[:, -1], dataset_right[:, -1]

                    # คำนวณค่า information gain
                    curr_info_gain = self.information_gain(y, left_y, right_y, "gini")

                    # อัปเดตการแบ่งที่ดีที่สุดหากค่าการเพิ่มข้อมูลสูงขึ้น
                    if curr_info_gain > max_info_gain:
                        best_split = {
                            "feature_index": feature_index,
                            "threshold": threshold,
                            "dataset_left": dataset_left,
                            "dataset_right": dataset_right,
                            "info_gain": curr_info_gain
                        }
                        max_info_gain = curr_info_gain

        # คืนค่าการแบ่งที่ดีที่สุด
        return best_split if "info_gain" in best_split else None
```

Random Forest Classifier Implementation

ในขั้นตอนนี้เราจะ implement Random Forest Classifier โดยมีการสร้างต้นไม้หลายต้น (Decision Trees) และใช้การทำนายผลจากแต่ละต้นไม้ร่วมกันเพื่อให้ผลลัพธ์สุดท้ายออกมาดีขึ้น

ภาพรวมการทำงานมี 2 ส่วน ที่สำคัญดังนี้ ¶

1. สร้าง Random Forest classifier: ฟังก์ชัน fit สร้างต้นไม้ตัดสินใจหลายต้นตามจำนวนที่กำหนด (n_estimators)
2. ทำนายค่า: ฟังก์ชัน predict ใช้การทำนายจากต้นไม้ทุกต้นในป่า และใช้วิธี majority vote เพื่อเลือกค่าที่เป็นค่าตอบสุดท้าย

```
In [ ]: # Random Forest Classifier Implementation
class RandomForestClassifierFromScratch:
    def __init__(self, n_estimators=10, max_depth=5, min_samples_split=2, max_features='sqrt'):
        self.n_estimators = n_estimators # จำนวนต้นไม้ที่จะสร้างใน Random Forest
        self.max_depth = max_depth # ความลึกสูงสุดของแต่ละต้นไม้
        self.min_samples_split = min_samples_split # จำนวนตัวอย่างข้อมูลขั้นต่ำที่ใช้ในการแบ่งโหนด
        self.max_features = max_features # จำนวนฟีเจอร์สูงสุด
        self.trees = []

    # ฟังก์ชันนี้ใช้ในการสร้าง Random Forest จาก Train set
    def fit(self, X, Y):
        self.trees = []
        for _ in range(self.n_estimators):
            X_sample, Y_sample = self._bootstrap_sampling(X, Y)
            tree = DecisionTreeClassifier(max_depth=self.max_depth, min_samples_split=self.min_samples_split)
            tree.fit(X_sample, Y_sample)
            self.trees.append(tree)

    # ฟังก์ชันนี้ใช้สำหรับการสุ่มข้อมูลแบบ bootstrap
    # bootstrap sampling ซึ่งเป็นการสุ่มข้อมูลแบบมีการทดแทน (replace=True)
    # หมายความว่าตัวอย่างข้อมูลเดิมสามารถถูกสุ่มซ้ำได้
    def _bootstrap_sampling(self, X, Y):
        n_samples = X.shape[0]
        indices = np.random.choice(n_samples, size=n_samples, replace=True)
        return X[indices], Y[indices]

    # ฟังก์ชันนี้ใช้สำหรับการทำนายค่าจากข้อมูลใหม่
    def predict(self, X):
        tree_predictions = np.array([tree.predict(X) for tree in self.trees])
        final_predictions = [self._majority_vote(tree_predictions[:, i]) for i in range(X.shape[0])]
        return final_predictions

    # ฟังก์ชันนี้ใช้สำหรับหาค่าตอบที่ได้จากการโหวตมากที่สุดจากผลการทำนายของต้นไม้หลายต้น
    def _majority_vote(self, predictions):
        return max(set(predictions), key=list(predictions).count)
```

ทำการแบ่งชุดข้อมูล train set และ test set

```
In [ ]: from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

ทำการสร้างโมเดล Random Forest

- กำหนดให้สร้างต้นไม้ 100 ต้น
- ความลึกสูงสุดของต้นไม้เป็น 12 ชั้น
- กำหนด sample ขั้นต่ำที่ใช้ในการแยกข้อมูล 3 ตัวอย่าง

```
In [ ]: # Create Random Forest Model
rf_classifier = RandomForestClassifierFromScratch(n_estimators=100, max_depth=12, min_samples_split=3)
rf_classifier.fit(X_train, Y_train)
```

```
In [ ]: # ทำการเก็บผลลัพธ์ที่คาดการณ์จากโมเดล Random Forest
Y_pred_rf = rf_classifier.predict(X_test)
```

ทำการคำนวณ accuracy ของโมเดล Random Forest

```
In [ ]: from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, Y_pred_rf)
print(f"Random Forest Model Accuracy: {accuracy}")
```

Random Forest Model Accuracy: 0.9126117179741807

สร้างและแสดง Classification Report

```
In [ ]: from sklearn.metrics import classification_report

report = classification_report(Y_test, Y_pred_rf)
print("Classification Report:\n", report)

Classification Report:
              precision    recall  f1-score   support

     0       0.78        0.61        0.69         95
     1       0.79        0.93        0.86         70
     2       1.00        1.00        1.00         78
     3       0.86        0.73        0.79        118
     4       1.00        1.00        1.00         69
     5       0.99        1.00        0.99         68
     6       0.85        0.91        0.88         66
     7       0.98        0.97        0.98         67
     8       0.90        0.96        0.93         74
     9       0.99        0.97        0.98         80
    10       0.95        0.99        0.97         80
    11       0.86        1.00        0.93         68
    12       0.95        1.00        0.97         74

 accuracy          0.91
 macro avg          0.92
 weighted avg       0.91
```

การบันทึกโมเดลที่สร้างขึ้น

```
In [ ]: import joblib
        joblib.dump(rf_classifier, 'model_now.joblib')

Out[16]: ['model_now.joblib']
```

2.3 การดึง Model มาใช้ร่วมกับ API ที่เขียน

การเขียน API ขึ้นมาเพื่อใช้สำหรับการดึง Model AI มาใช้ผ่าน Cloud โดยใน Code เป็นการใช้ flaskAPI ในโค้ดประกอบด้วยการสร้าง instance ของ FlaskAPI แล้วทำการโหลด model และสร้าง class ของ model Random forest ที่สร้างขึ้นเอง และสร้าง Route ในการรับ input จาก user แล้วนำค่า input ไป encoder แล้วเข้าโมเดลให้ทำนาย ประเภทชีสออกมาแล้วทำการ response กลับไปยัง user

(เปิดดู Code ได้ที่

https://drive.google.com/drive/folders/17eazG5SmTlL4ZnRc6ZH1bxjQTUZp92hW?usp=drive_link)

```
# โหลดโมเดลที่ได้รับการฝึกมา
loaded_rf_classifier = joblib.load('model_now.joblib')

def get_closest_match(value, valid_values):
    # print(valid_values)
    if isinstance(value, str): # ตรวจสอบว่าเป็น string หรือไม่
        # ตรวจสอบ exact match ก่อน
        if value in valid_values:
            return value

        # ตรวจสอบกรณีสลับตำแหน่งคำ เช่น 'United Kingdom, Scotland' กับ 'Scotland, United Kingdom'
        for valid_value in valid_values:
            if set(valid_value.split(', ')) == set(value.split(', ')):
                return valid_value

        # ใช้ fuzzy matching กับทุกคำที่อยู่ใน valid_values
        matches = process.extract(value, valid_values, limit=None)

        # สืบเสาะค่าที่ดีที่สุด
        best_match = None
        best_score = 0

        for match, score in matches:
            #print(f'Matching {value} with candidate {match}, score: {score}')
            if score > best_score:
                best_score = score
                best_match = match

        # ปรับเกณฑ์คะแนนให้สูงขึ้นเพื่อความแม่นยำมากขึ้น
        return best_match if best_score >= 90 else None

    return None # ถ้าไม่ใช่ string ให้คืนค่า None
```



```
def fill_missing_values(encoded_input, df):
    # Convert to NumPy array, handling None
    encoded_input = np.array([x if x is not None else np.nan for x in encoded_input], dtype=float)

    # Check for NaN values
    if np.any(np.isnan(encoded_input)):
        # Fill missing values using mode of the corresponding column
        for i in range(len(encoded_input)):
            if np.isnan(encoded_input[i]):
                # Get mode of the column from the original DataFrame
                mode_values = df.iloc[:, i].mode() # Get mode of the column
                print(mode_values)
                if not mode_values.empty: # Ensure there is at least one mode value
                    encoded_input[i] = mode_values[0] # Use the first mode value
                print(encoded_input)

    return encoded_input
```

```
# เริ่มต้นเขียน main function
@app.route('/predict', methods=['POST', 'OPTIONS'])
def predict():
    global shared_encoded_input, shared_prediction

    if request.method == 'OPTIONS':
        response = app.make_default_options_response()
        headers = response.headers
        headers['Access-Control-Allow-Origin'] = 'http://127.0.0.1:5500'
        headers['Access-Control-Allow-Methods'] = 'POST, OPTIONS'
        headers['Access-Control-Allow-Headers'] = 'Content-Type'
        return response, 200

    try:
        # ดึงข้อมูล JSON จาก request
        data = request.get_json(force=True)
        input_data = data['input'] # ต้องเป็น list ของ string หรือ null
        print(input_data)

        # เริ่มการแปลง input โดยไม่ static mapping และ fuzzy matching
        encoded_input = []
        index_key_mapping = ['milk', 'country', 'type', 'fat_content', 'texture', 'rind', 'color', 'flavor', 'aroma', 'vegetarian', 'vegan']

        # คู่มือกับการจัดการความสัมพันธ์ index_key_mapping
        for i, value in enumerate(input_data[0]):
            if i < len(index_key_mapping):
                col_name = index_key_mapping[i] # แปลงค่าชื่อ col_name จาก static_mapping โดยไม่ index_key_mapping
                if value is None: # ถ้าเป็นค่า null ไม่ใส่ค่า None
                    encoded_input.append(None)
                elif any(key.lower() == value.lower() and len(key) == len(value) for key in en_static_mapping[col_name].keys()):
                    exact_key = next(key for key in en_static_mapping[col_name].keys() if key.lower() == value.lower() and len(key) == len(value))
                    encoded_input.append(en_static_mapping[col_name][exact_key])
                else: # ใช้ fuzzy matching หากไม่เจอ exact match
                    closest_match = get_closest_match(value, en_static_mapping[col_name].keys())
                    if closest_match:
                        encoded_input.append(en_static_mapping[col_name][closest_match])
                    else:
                        encoded_input.append(None) # ถ้าไม่เจอ match ใดๆ ไม่ใส่ None
```

```
# ตรวจสอบ encoded_input
print(encoded_input)
shared_encoded_input = encoded_input

# Convert to NumPy array
encoded_input = np.array(encoded_input, dtype=object).reshape(1, -1)

# เติมค่าที่ขาดหาย
filled_input = fill_missing_values(encoded_input[0], df)

# แปลง filled_input เป็น NumPy array
filled_input = np.array(filled_input, dtype=float).reshape(1, -1)

# ใช้ model random forest ในการทำนาย
predictions = loaded_rf_classifier.predict(filled_input)
shared_prediction = predictions
print(predictions)

# Convert predictions ให้เป็นชื่อ class
class_names = ['Cheddar', 'Blue', 'Brie', 'Pecorino', 'Gouda', 'Parmesan', 'Camembert', 'Feta',
               'Cottage', 'Pasta filata', 'Swiss Cheese', 'Mozzarella', 'Tomme']
predictions_index = [3, 0, 1, 10, 6, 8, 2, 5, 4, 9, 11, 7, 12]
predicted_class_names = [class_names[predictions_index.index(pred)] for pred in predictions]

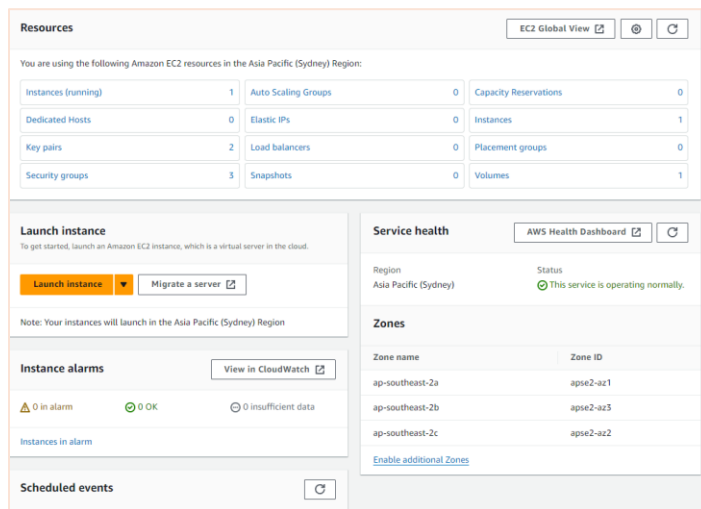
return jsonify({'prediction': predicted_class_names})

except Exception as e:
    return jsonify({'error': str(e)}), 500
```

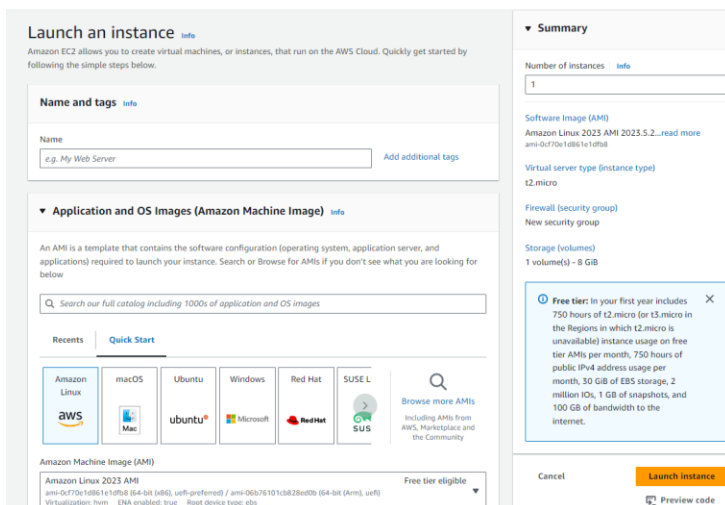
2.4 การนำ Model ML (Machine Learning) ขึ้น Cloud AWS EC2 โดยใช้ Flask API



ทำการเข้าเว็บ <https://aws.amazon.com/th/> เพื่อทำการสมัครใช้บริการคลาวด์ของ AWS จากนั้นทำการค้นหา EC2 ที่ search bar แล้วคลิกเข้ามาที่หน้าบริการของ EC2 บริเวณกลางซ้ายของหน้าจะเห็นปุ่ม “Launch Instance” ให้ทำการคลิกเข้าไปเพื่อทำการสร้าง Instance สำหรับ Deploy Model ML เข้าไป



การสร้าง Instance



1. ใส่ชื่อ Instance
2. เลือกระบบปฏิบัติการ Ubuntu Server 22.04 LTS (HVM)
3. Instance Type ให้เลือก t2.micro
4. Key pair (login) ให้กด Create new key pair
 - ใส่ชื่อ key pair
 - เลือก RSA
 - เลือกไฟล์นามสกุล .ppk
 - **Create key pair**
5. Network Setting

The screenshot shows the 'Network settings' section of the AWS console. It includes fields for 'Network' (vpc-0884f6fd339e7fab9) and 'Subnet' (No preference). The 'Auto-assign public IP' is set to 'Enable'. Under 'Firewall (security groups)', the 'Create security group' option is selected. Below this, it states that a new security group named 'launch-wizard-3' will be created with three rules: 'Allow SSH traffic from Anywhere', 'Allow HTTPS traffic from the internet', and 'Allow HTTP traffic from the internet'.

▼ Network settings [Info](#) Edit

Network | [Info](#)
vpc-0884f6fd339e7fab9

Subnet | [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)
Enable

[Additional charges apply](#) when outside of [free tier allowance](#)

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-3' with the following rules:

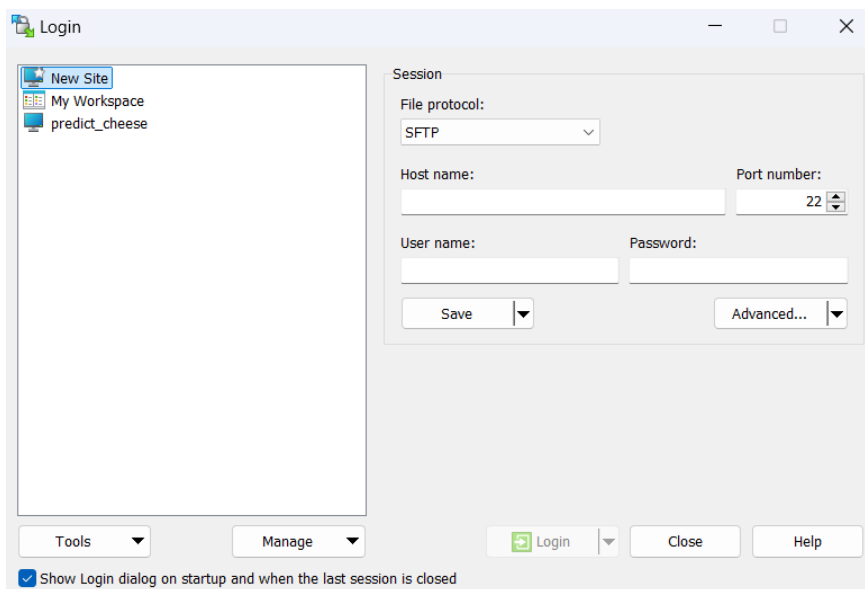
- ☒ Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
- ☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
- ☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

6. Configure storage ทำการกำหนดพื้นที่จัดเก็บข้อมูลตามสมควร เบื้องต้นใช้ 8 GiB
7. จากนั้นทำการ “Launch Instance”

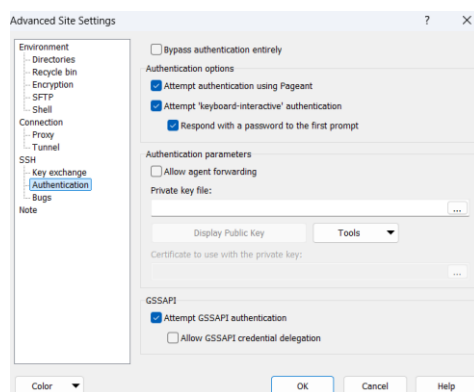
Upload Model to Instance Storage



ใช้โปรแกรม WinSCP ในการอัปโหลดไฟล์ API และ ML Model และไฟล์ต่างๆที่เกี่ยวข้องกับการใช้ AI จากหัวข้อที่แล้วซึ่งเราได้ไฟล์ key pair ที่เป็นนามสกุล .ppk เราจะนำมาเป็นกุญแจในการ Access เข้า storage ของ instance

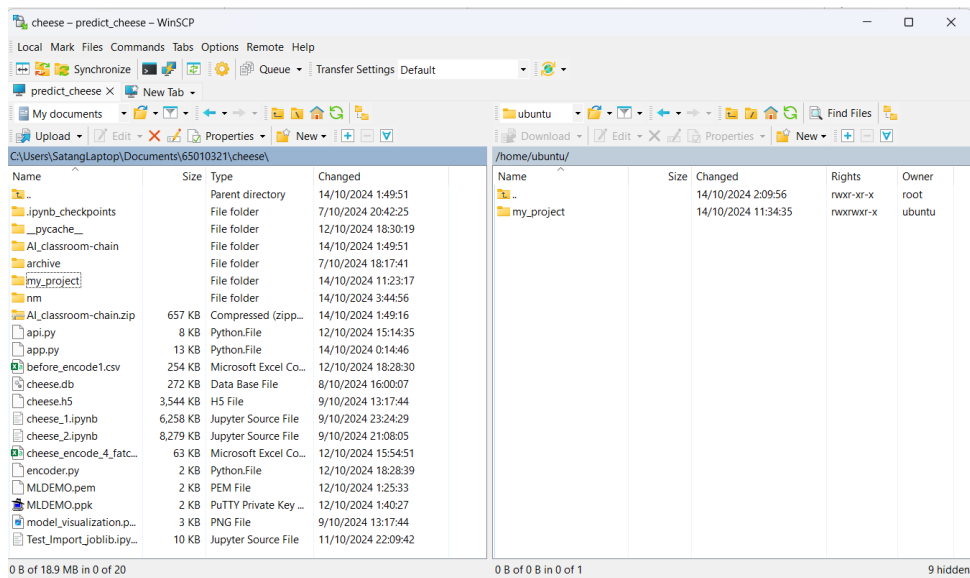


1. Host Name : ในใต้ Public IPv4 ของ Instance ที่สร้างขึ้น
2. Username : เนื่องจากเราใช้ Linux ของ Ubuntu ชื่อ Username ของเราจึงเป็น ubuntu
3. คลิกไปที่ “Advance” เลือกรหัสชื่อ Authentication ส่วนของ Private key file ให้เลือกไฟล์ที่ได้มาจากการสร้าง Key pair (.ppk)



จากนั้นกด OK แล้วก็จะสามารถ Login เข้าสู่ storage ของ Instance ได้

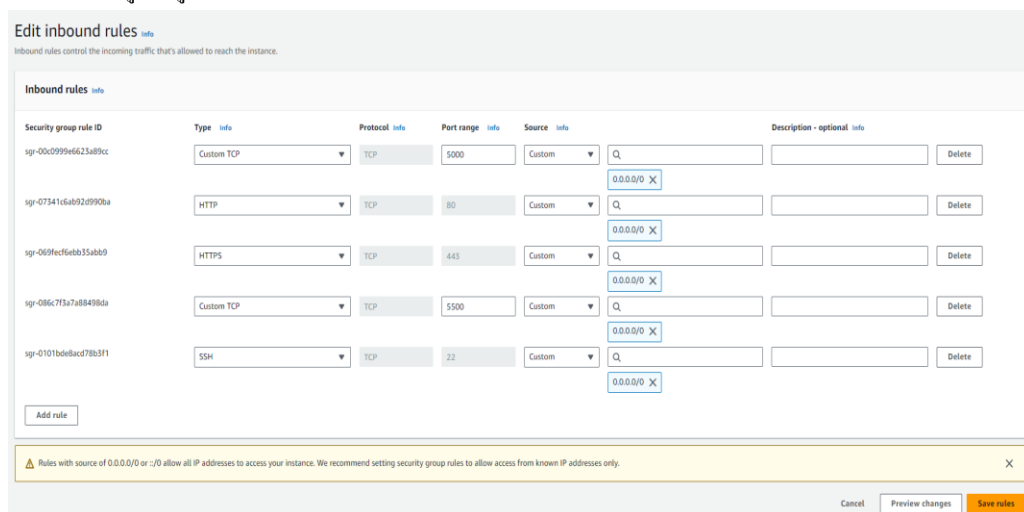
4. จากนั้นให้ทำการลากหรืออัปโหลดไฟล์เตอร์ที่เก็บไฟล์ของ Model AI และ API ไปยัง storage ของ Instance ดังรูป



การกำหนด Security Group ของ Instance

เนื่องจากว่า EC2 มีการกำหนด Security Group เพื่อให้ตัวเว็บไซต์ที่เราทำขึ้น หรือ การส่ง API Tester อย่างเช่น POSTMAN สามารถส่งค่าเข้ามาใน Instance ผ่าน API ได้

1. ให้ไปที่ EC2 Dashboard -> Security Group
2. เลือก Group Security ของ Instance ที่เราสร้างไว้ -> คลิกไปที่ลิงก์ Security ID
3. ให้มาดูส่วนของ Inbound Rules -> คลิกที่ Edit Inbound Rules จากนั้นให้ทำการ อนุญาต PORT 5000 เนื่องจาก Flask API จะใช้ PORT 5000 ในการรับ-ส่งข้อมูล ดังรูป (สามารถเพิ่ม Inbound Rules ตามการใช้งานจริงได้เลย)

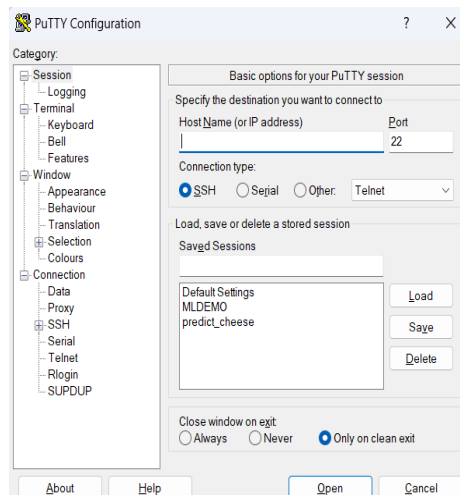


4. จากนั้นกด “Save rules”

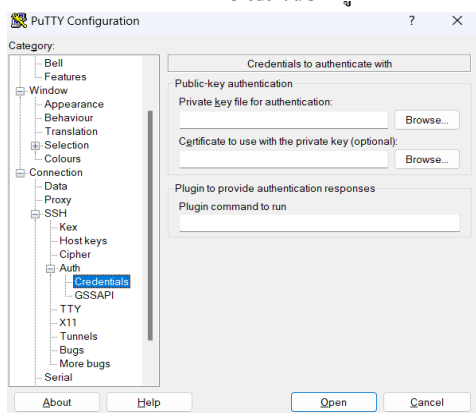
การใช้ PuTTY ในการ access เข้า Ubuntu Linux



ในการ access เข้าสู่ Linux นั้น สามารถทำได้หลายวิธี ซึ่งหนึ่งในวิธีที่ได้รับความนิยมสูงเนื่องจากมีความสะดวกสบายจากการリモテเข้าไปทำงานต่างๆในระบบ นั่นคือ PuTTY ที่เป็นโปรแกรมสำหรับการเข้า SSH ของ Instance เมื่อดาวน์โหลด PuTTY แล้วสามารถทำตามขั้นตอนตามด้านล่างได้เลย

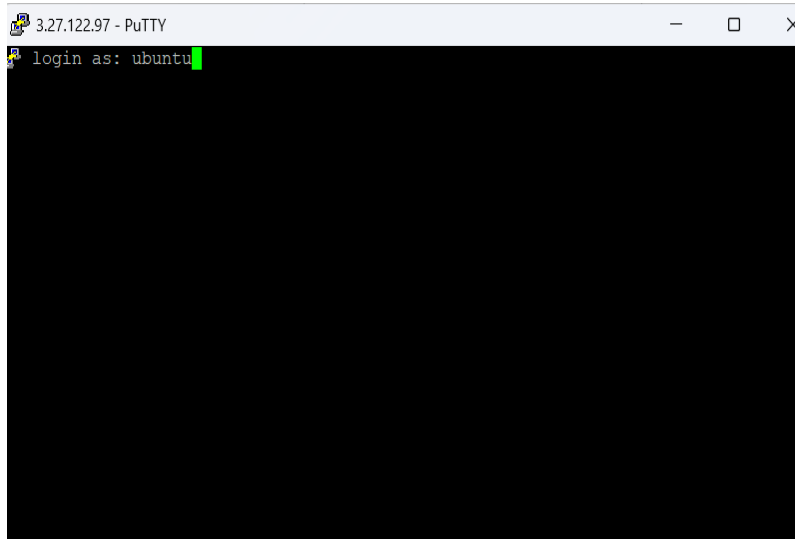


1. Host Name ให้ใส่ Public IPv4 ของ Instance
2. จากนั้นแถบด้านซ้ายจะเห็นหัวข้อ SSH ให้คลิกปุ่ม + ข้างคำ SSH ลงมา -> จากนั้นจะเห็นคำว่า Auth ที่มีเครื่องหมาย + อยู่ด้านข้างเช่นกัน ให้กดลงมา แล้วคลิกไปที่ Credentials ดังรูป

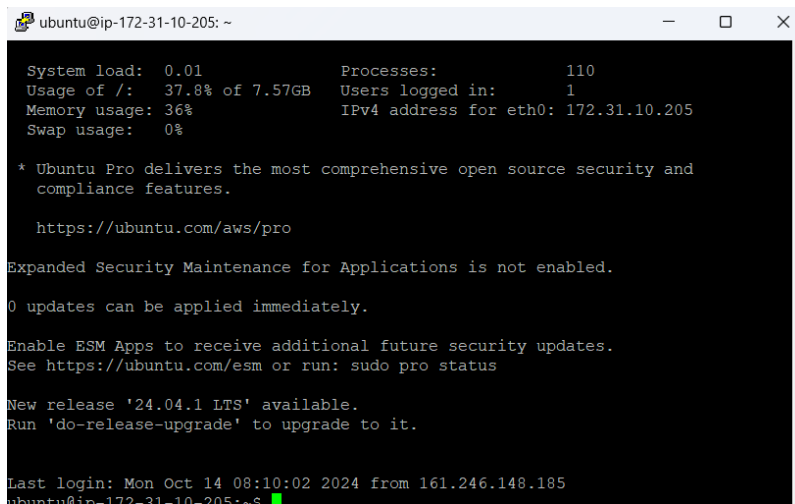


3. จากนั้นส่วนของ Private key file ให้อัปโหลดไฟล์ Key Pair ที่เป็นนามสกุล .ppk ที่เราได้มาจากก่อนหน้านี้ จากนั้นให้กลับไปหน้า Session เพื่อทำการ login เข้า Ubuntu

4. จากนั้นก็จะขึ้นมาให้ทำการ login -> ใส่คำว่า ubuntu เพื่อทำการ login ดังรูป



ก่อนทำการ Login



หลังจาก Login สำเร็จ

การรันไฟล์ API ที่อยู่บนคลาวด์ผ่าน Putty

ในการจะรัน Flask API ที่อยู่บนคลาวด์ได้นั้น จำเป็นต้องมีการเตรียมระบบเล็กน้อยก่อน โดยใช้คำสั่งต่อไปนี้

```
sudo apt update
```

```
sudo apt upgrade
```

```
sudo apt install python3 python3-pip -y
```

หลังจากที่รันคำสั่งด้านบนเสร็จหมดแล้ว ให้ทำการใช้ command `pip install` เพื่อดาวน์โหลด Library ทั้งหมดที่เราจะใช้งาน ในการรัน API หลังจากเมื่อดาวน์โหลด Library ที่ใช้ทั้งหมดจนเสร็จสมบูรณ์ ให้ทำขั้นตอนด้านล่างต่อไปนี้

1. ทำการ `cd` เข้าสู่โฟลเดอร์ที่มีไฟล์ API ของเรายู่
`cd my_project` -> สามารถเปลี่ยนตาม path และชื่อของโฟลเดอร์ได้
2. จากนั้นให้ใช้คำสั่ง `ls` เพื่อดูรายชื่อไฟล์ที่อยู่ในโฟลเดอร์ทั้งหมด
3. ใช้คำสั่ง `python3 app.py` เพื่อรัน API
หมายเหตุ : `app.py` อาจไม่ใช่ชื่อของ API ที่ต้องการรัน ดังนั้นให้รันตามชื่อไฟล์ API ที่อัปโหลดไป

จากขั้นตอนก่อนหน้าจะทำให้สามารถรัน API ได้ผลลัพธ์ตามรูปต่อไปนี้

```
warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.26.5.179:5000
Press CTRL+C to quit
```

แต่ว่าการรันด้วย Python ตรงๆแบบนี้ ณ จุดๆหนึ่ง session การทำงานของโค้ดที่จะหยุดและหายไป เพื่อให้ API ของเราสามารถทำงานตลอดระยะเวลาที่ Instance ทำงานอยู่ จะสามารถทำได้ในขั้นตอนต่อไป

การรัน API ตลอดเวลาไม่ให้เกิดการ Aborted หรือ Session closed

ในการจัดการ Flask API บนเซิร์ฟเวอร์ Ubuntu เพื่อให้มั่นใจว่าแอปพลิเคชันสามารถรันได้อย่างต่อเนื่องและหลีกเลี่ยงปัญหาการหยุดทำงาน (Aborted) หรือการปิดเซสชัน (Session closed) เมื่อทำงานบนระบบคลาวด์หรือเซิร์ฟเวอร์ จำเป็นต้องตั้งค่าให้ระบบสามารถจัดการการเริ่มต้นและการดูแลกระบวนการทำงานของ API โดยอัตโนมัติผ่านการใช้ `systemd` ซึ่งเป็นระบบจัดการกระบวนการใน Linux ด้วยการสร้างและจัดการไฟล์ service จะช่วยให้มั่นใจว่า API จะเริ่มทำงานใหม่โดยอัตโนมัติหากเกิดการขัดข้อง และยังสามารถควบคุมได้ง่ายในระยะยาว

ให้ทำตามขั้นตอนต่อไปนี้

1. เปิดไฟล์และแก้ไขการตั้งค่า Systemd
`sudo nano /etc/systemd/system/flaskapp.service`
2. ไฟล์ `flaskapp.service` มีโค้ดดังต่อไปนี้

```
[Unit]
```

```
Description=Flask API
```

```
After=network.target
```

```
[Service]
```



```
User=ubuntu
WorkingDirectory=/home/ubuntu/my_project
ExecStart=/usr/bin/python3 /home/ubuntu/my_project/app.py
```

```
[Install]
WantedBy=multi-user.target
```

*****หมายเหตุ***** path ของไฟล์ที่ต้องการตั้งค่าให้รันตลอดเวลาสามารถเปลี่ยนได้ตาม directory ของไฟล์ API

3. โหลดการตั้งค่าใหม่เข้าสู่ system

```
sudo systemctl daemon-reload
```
4. เริ่มต้น Flask API service

```
sudo systemctl start flaskapp
```
5. เปิดใช้งาน Flask API ให้เริ่มต้นโดยอัตโนมัติเมื่อระบบรีบูต

```
sudo systemctl enable flaskapp
```

ตรวจสอบสถานะของ Flask API service

```
sudo systemctl status flaskapp
```

2.5 ในส่วนของฟังก์ชัน What Cheese จะทำให้ Model AI และ API อีกตัวหนึ่ง

1. การเตรียม Dataset และแบ่งอัตราส่วนข้อมูล

ได้ทำการเขียน Code เพื่อดึงรูปภาพจาก Google ตามประเภทชีสต่างๆ ประเภทละ 200 รูป หลังจากนั้นทำการแบ่งอัตราส่วนข้อมูล Train 70 % และ Test 30% (เปิดดู Code และ Dataset ได้ที่

<https://drive.google.com/drive/folders/1PgMgU5uJw8yKy2RMwkK0wLNVNY-uTZEa?usp=sharing>)

2. กระบวนการ Train และ Test Model AI

ใช้เว็บไซต์ Teachable Machine ในการ Train Model AI ขึ้นมา โดยแบ่งประเภทชีสเป็นทั้งหมด 20 class ได้แก่ Blue, Brie, Caciotta, Camembert, Cheddar, Cornish cheese, Cottage, Feta, Gouda, Havarti cheese, Italian Cheese, Monterey Jack, Mozzarella, Parmesan, Pasta filata, Pecorino, Raclette cheese, Saint-

paulin cheese, Swiss Cheese และ Tomme หลังจาก Train Model เสร็จทำการแปลง Model ให้เป็น keras.h5 เพื่อนำไปใช้ต่อ

3. ทดลองใช้ Model keras.h5 บน jupyter notebook

นำ Model keras.h5 ที่ได้จาก Teachable Machine มาทดสอบการใช้งานบน jupyter notebook (เปิดดู Code และ Dataset ได้ที่ https://drive.google.com/drive/folders/10Ts4DQtGjApO0gNsd-Rs21YGj3DsfrMN?usp=drive_link)

```
[26]: !pip install tensorflow==2.12.0

[28]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
print(tf.__version__)

2.12.0

[29]: # โหลดโมเดลจากไฟล์ .h5
model = tf.keras.models.load_model(r'C:\Users\Bam\Documents\65010556\TestModel_ImageClassifierCheese\keras_model.h5')

# โหลดรายชื่อคลาสจากไฟล์ labels.txt
with open(r'C:\Users\Bam\Documents\65010556\TestModel_ImageClassifierCheese\labels.txt', 'r') as f:
    class_labels = [line.strip() for line in f]

print("Model and labels loaded successfully!")

WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
Model and labels loaded successfully!

[30]: # กำหนด path ของภาพที่ต้องการทำนาย
img_path = r'C:\Users\Bam\Documents\65010556\TestModel_ImageClassifierCheese\image\Test\Blue\Blue cheese_162.jpg' # แทนที่ด้วย path ของภาพที่คิดต้องการ
img = image.load_img(img_path, target_size=(224, 224)) # ปรับขนาดตามที่โมเดลต้องการ
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) # เพิ่มมิติสำหรับ batch
img_array = img_array / 255.0 # Normalize ค่า

[31]: # ทำการทำนาย
predictions = model.predict(img_array)
# ดึงความมั่นใจ (confidence) ของแต่ละคลาส
confidence_scores = predictions[0]

# แสดงชื่อคลาสพร้อมกับความมั่นใจของแต่ละคลาสในรูปแบบเปอร์เซ็นต์
print("Confidence scores for each class:")
for label, score in zip(class_labels, confidence_scores):
    print(f"{label}: {score * 100:.2f}%")

1/1 [=====] - 1s 531ms/step
Confidence scores for each class:
Blue: 98.11%
Brie: 0.00%
Caciotta: 0.65%
Camembert: 0.00%
Cheddar: 0.00%
Cornish cheese: 0.00%
Cottage: 0.14%
Feta: 0.01%
Gouda: 0.00%
Havarti cheese: 0.00%
Italian Cheese: 0.06%
Monterey Jack: 0.00%
Mozzarella: 0.00%
Parmesan: 0.00%
Pasta filata: 0.00%
Pecorino: 0.39%
Raclette cheese: 0.00%
Saint-paulin cheese: 0.00%
Swiss Cheese: 0.00%
Tomme: 0.63%
```

```
[34]: # ดึงค่าความมั่นใจ (confidence) ของคลาสที่โมเดลทำนายได้ที่ดีที่สุด
predicted_index = np.argmax(predictions[0])
predicted_class = class_labels[predicted_index]
confidence = predictions[0][predicted_index]

# แสดงผลลัพธ์
print(f"Predicted Class: {predicted_class}")
print(f"Confidence: {confidence * 100:.2f}%")

# แสดงภาพพร้อมชื่อคลาสและความมั่นใจ
plt.imshow(image.load_img(img_path))
plt.title(f"Predicted: {predicted_class} ({confidence * 100:.2f}%")
plt.axis('off')
plt.show()

Predicted Class: Blue
Confidence: 98.11%
```



4. การดึง Model มาใช้ ร่วมกับ API ที่เขียนขึ้นมา

จากนั้นทำการเขียน API ขึ้นมาเพื่อใช้สำหรับการดึง Model AI มาใช้ผ่าน Cloud โดยใน Code จะประกอบ 1.การสร้าง instance ของ FastAPI 2.การโหลด Model และ Class 3.การเตรียมข้อมูลจากรูปภาพก่อนส่งเข้า Model 4.การสร้าง Route สำหรับ predict รูปภาพที่ผู้ใช้ส่งมา (เปิดดู Code ได้ที่

https://drive.google.com/drive/folders/1eBBpqyLvIsxxQFHOkEF-h_-4kxqs-Nff?usp=drive_link)

```
1 from fastapi import FastAPI, File, UploadFile
2 from fastapi import Security, HTTPException, Depends
3 from fastapi.security.api_key import APIKeyHeader
4 from fastapi.middleware.cors import CORSMiddleware
5
6 import uvicorn
7 from tensorflow.keras.models import load_model
8 import numpy as np
9 from PIL import Image
10 import io
11
12 # สร้าง instance ของ FastAPI
13 app = FastAPI()
14 app.add_middleware(
15     CORSMiddleware,
16     allow_origins=["*"],
17     allow_credentials=True,
18     allow_methods=["*"],
19     allow_headers=["*"],
20 )
21
22 # โหลดโมเดลของคุณ และโหลด class
23 model = load_model('./models/keras_model.h5')
24 with open(r'./models/labels.txt', 'r') as f:
25     class_labels = [line.strip() for line in f]
26
```

```

27 # ฟังก์ชันสำหรับการเตรียมข้อมูลจากภาพก่อนส่งเข้าโมเดล
28 def preprocess_image(image: Image.Image):
29     # แปลงภาพให้เป็น RGB เสมอ
30     if image.mode != 'RGB':
31         image = image.convert('RGB')
32     # ปรับขนาดของรูปภาพให้ตรงกับขนาด input ของโมเดล
33     image = image.resize((224, 224)) # ตัวอย่าง, หากโมเดลต้องการขนาด 224x224
34     image = np.array(image) / 255.0 # แปลงเป็น numpy array และปรับค่าความเข้มของสีให้อยู่ในช่วง 0-1
35     image = np.expand_dims(image, axis=0) # เพิ่มมิติใหม่เพื่อให้ตรงกับ input ของโมเดล (batch size)
36     return image
37
38 # Route สำหรับการพยากรณ์จากภาพที่ผู้ใช้ส่งมา
39 @app.post("/predict")
40 async def predict_image(file: UploadFile = File(...)):
41     try:
42         # ตรวจสอบชนิดไฟล์
43         if file.content_type not in ['image/jpeg', 'image/png']:
44             return {"error": "Invalid file type."}
45
46         # อ่านไฟล์รูปภาพ
47         image_bytes = await file.read()
48         image = Image.open(io.BytesIO(image_bytes))
49
50         # เตรียมข้อมูลรูปภาพ
51         processed_image = preprocess_image(image)
52
53         # ส่งรูปภาพไปยังโมเดลเพื่อทำการพยากรณ์
54         predictions = model.predict(processed_image)
55         predicted_class = np.argmax(predictions[0]) # หาประเภทที่มีค่าเป็นไป้สูงสุด
56
57         # ดึงค่าความมั่นใจ (confidence) ของคลาสที่โมเดลทำนายได้ดีที่สุด
58         predicted_index = np.argmax(predictions[0])
59         predicted_class = class_labels[predicted_index]
60         confidence = predictions[0][predicted_index]
61         print(f"Predicted Class: {predicted_class}")
62         print(f"Confidence: {confidence * 100:.2f}%")
63
64         # ส่งผลลัพธ์กลับไปในรูปแบบ JSON
65         return {
66             "Predicted": predicted_class,
67             "Confidence": f"{confidence * 100:.2f}%"
68         }
69
70     except Exception as e:
71         raise HTTPException(status_code=500, detail=str(e))
72
73 if __name__ == "__main__":
74     uvicorn.run(app, host="0.0.0.0", port=8000)

```

5. การสร้าง Instances บน AWS EC2 Cloud

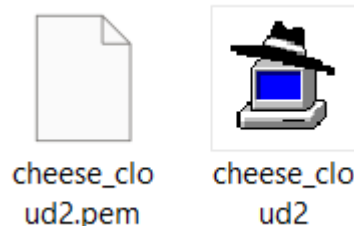
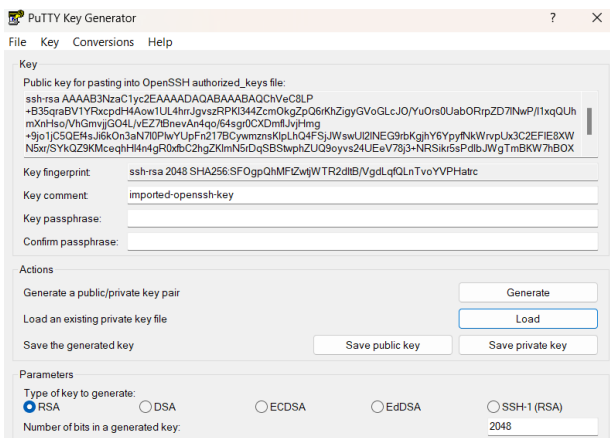
หลังจากที่ AWS Management Console ให้เลือกใช้บริการ EC2 และทำการสร้าง Instance โดยได้มีการกำหนดชื่อเป็น cheese_cloud2 ในส่วนของ Application and OS Images ให้เลือกใช้บริการของ Ubuntu Server 22.04 กำหนด Instance type เป็นแบบ t2.micro และสุดท้ายทำการสร้าง Key pair ใหม่ให้อยู่ในนามสกุล .pem

Instance summary for i-0aa92d83ebf031e7f (cheese_cloud2) Info Updated less than a minute ago		
Instance ID i-0aa92d83ebf031e7f (cheese_cloud2)	Public IPv4 address 3.107.26.166 open address	Private IPv4 addresses 172.31.0.246
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-107-26-166.ap-southeast-2.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-0-246.ap-southeast-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-0-246.ap-southeast-2.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 3.107.26.166 [Public IP]	VPC ID vpc-08c978c70777c16e4	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0100af2b585c9f78f	
IMDSv2 -	Instance ARN -	

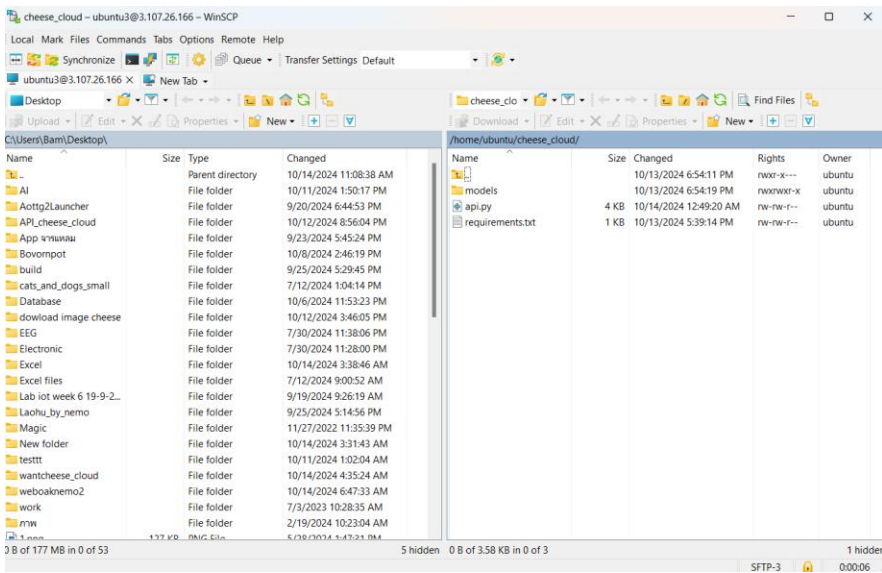
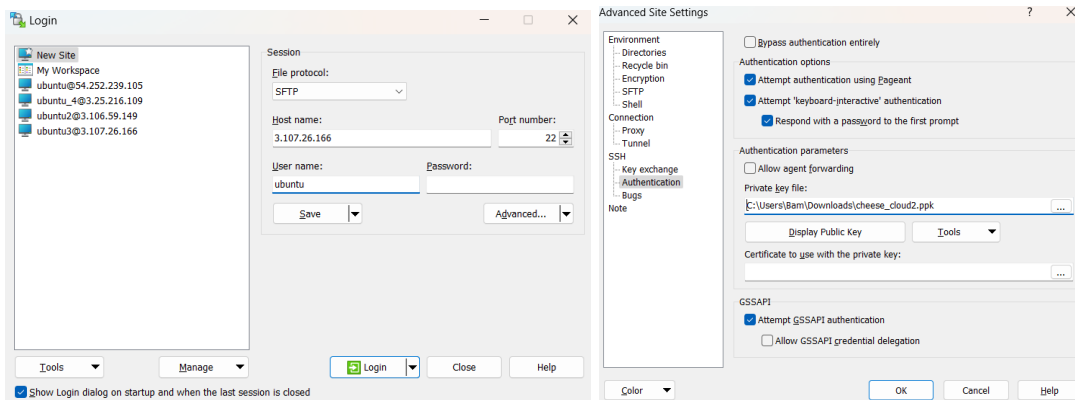
6. การ Deploy Model AI ขึ้น Cloud

ในส่วนนี้ได้ใช้โปรแกรมช่วยทั้งหมด 3 โปรแกรม ได้แก่ PuTTY PuTTYgen และ WinSCP

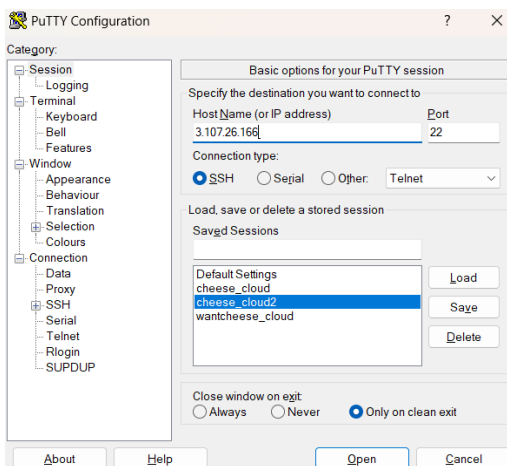
1.เปิด PuTTYgen และ load ไฟล์ Key pair นามสกุล .pem จากนั้น Save private key เพื่อนำ key ตรงนี้ไปใช้ในการ เข้าถึงไฟล์ต่างๆบน Cloud EC2 ที่เราสร้างไว้ก่อนหน้านี้ ซึ่งไฟล์ Key ใหม่มีนามสกุลคือ .ppk



2.เปิด WinSCP เพื่อเพิ่มไฟล์ Model AI และ API ที่เราทำไว้เข้าไปบน Cloud โดยกำหนด Host name เป็น Public IPv4 address ส่วน User name เขียนว่า ubuntu ส่วนpassword ให้เลือก Advanced -> SSH -> Authentication -> Private key file ให้เลือกเป็นไฟล์ .ppk ที่เรา gen ขึ้นมา จากนั้นทำการใส่โฟลเดอร์ API ที่ประกอบด้วย Model api.py ของเราเข้าไป



3.เปิด PuTTY เพื่อทำการ setup และเปิดการทำงานของ Cloud ทำการติดตั้ง library ต่างๆ และเปิดการใช้งาน Cloud ซึ่งใน requirements ประกอบไปด้วย fastapi, uvicorn, tensorflow-cpu==2.12.0 --no-cache-dir , pillow , numpy, PTL, flask-cors



```
ubuntu@ip-172-31-0-246:~$ sudo apt update
sudo apt upgrade
sudo apt install python3 -y
sudo apt install python3-pip -y
```

```
pip install -r requirements.txt
```

หลังจากติดตั้ง library ครบทั้งหมด ให้ใช้ systemd เพื่อให้ API ทำงานตลอดเวลา

สร้างไฟล์ Service

```
sudo nano /etc/systemd/system/api.service
```

เพิ่มเนื้อหาในไฟล์ Service

```
[Unit]
Description=API Service
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/home/ubuntu/cheese_cloud # เปลี่ยนให้ตรงกับที่
ExecStart=/usr/bin/python3 /home/ubuntu/cheese_cloud/api.py
Restart=always

[Install]
WantedBy=multi-user.target
```

ตั้งค่าให้ service ทำงานอัตโนมัติ

```
sudo systemctl daemon-reload
sudo systemctl start api.service
sudo systemctl enable api.service
```

เมื่อ check status จะบอกว่า active (running) ถือว่าทำงานได้

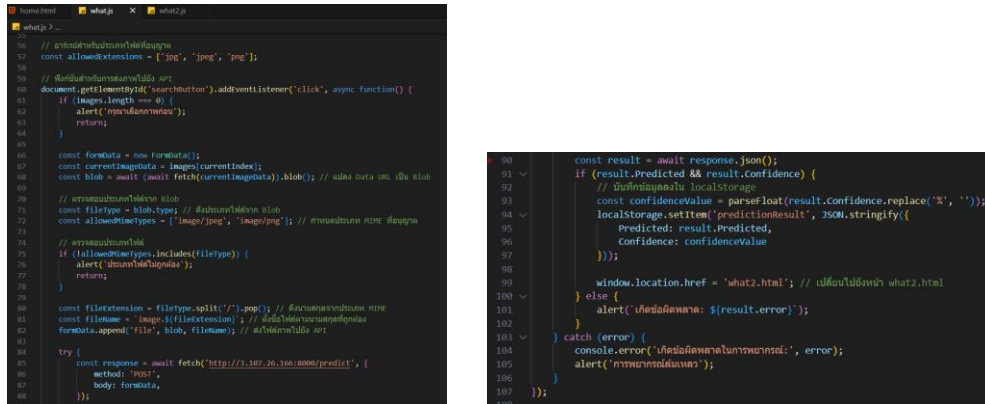
```
ubuntu@ip-172-31-0-246: ~
ubuntu@ip-172-31-0-246:~$ sudo systemctl status api.service
● api.service - API Service
   Loaded: loaded (/etc/systemd/system/api.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-10-13 17:51:24 UTC; 11h ago
     Main PID: 12236 (python3)
        Tasks: 6 (limit: 1130)
       Memory: 356.2M
          CPU: 1min 15.620s
       CGroup: /system.slice/api.service
               └─12236 /usr/bin/python3 /home/ubuntu/cheese_cloud/api.py

Oct 14 02:42:19 ip-172-31-0-246 python3[12236]: [145B blob data]
Oct 14 02:42:19 ip-172-31-0-246 python3[12236]: Predicted Class: Blue
Oct 14 02:42:19 ip-172-31-0-246 python3[12236]: Confidence: 49.54%
Oct 14 02:42:19 ip-172-31-0-246 python3[12236]: INFO:      161.246.151.120:19865 - "POST /predict HTTP/1.1" 200 OK
Oct 14 03:22:14 ip-172-31-0-246 python3[12236]: INFO:      103.203.57.20:36878 - "GET / HTTP/1.1" 404 Not Found
Oct 14 03:46:46 ip-172-31-0-246 python3[12236]: INFO:      87.236.176.87:50909 - "GET / HTTP/1.1" 404 Not Found
Oct 14 04:16:34 ip-172-31-0-246 python3[12236]: [145B blob data]
Oct 14 04:16:34 ip-172-31-0-246 python3[12236]: Predicted Class: Blue
Oct 14 04:16:34 ip-172-31-0-246 python3[12236]: Confidence: 96.91%
Oct 14 04:16:34 ip-172-31-0-246 python3[12236]: INFO:      161.246.149.186:10631 - "POST /predict HTTP/1.1" 200 OK
ubuntu@ip-172-31-0-246:~$
```

7. การเรียกใช้ AI จากบนเว็บไซต์

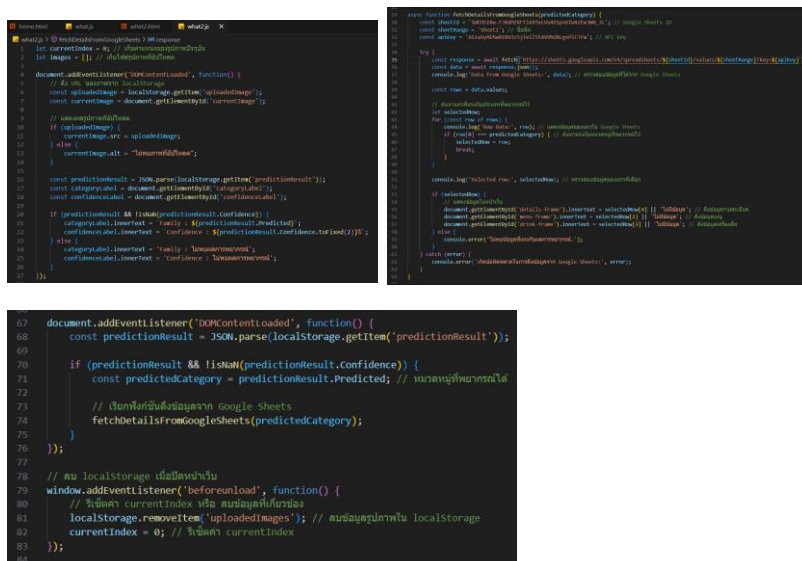
ในส่วนหน้า What แสดงถึงการส่งภาพที่มีการ upload จากเว็บไซต์ ไปยัง API เพื่อให้ AI predict ประเภทของชีส และบันทึกผลลัพธ์จากการ predict ไว้ใน localStorage เพื่อนำไปแสดงผลต่อในหน้า What2

(เปิดดู code ได้ที่ https://drive.google.com/file/d/1DIhuE-ftC4t1Zl6Cx9XdkGLTPwufeZ1V/view?usp=drive_link)



การแสดงผลในส่วนหน้า What 2 จะแสดงถึงการนำผล predict ที่ได้จาก AI รวมถึงนำประเภทชีสที่ได้ ไปดึงข้อมูล Detail, Menu และ Drink จากDataset ที่ทำบนGoogle Sheet เพื่อให้แสดงผลรายละเอียด เมนู เครื่องดื่ม ที่เหมาะกับประเภทชีสนั้นๆ

(เปิดดู code ได้ที่ https://drive.google.com/file/d/1hy8T0RUB2UAjOSSvDeKTqCmO-sk5f15p/view?usp=drive_link)



2.6 การทำ Web Hosting

ให้ทำการสมัครสมาชิกของ aws ตามลิงค์เว็บไซต์ต่อไปนี้ <https://us-east-1.console.aws.amazon.com>

จากนั้นให้ทำการเข้าไปที่ S3 แล้วกดปุ่ม create bucket

AWS Region
US East (N. Virginia) us-east-1

Bucket type **Info**

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name **Info**

thiencharoen3

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

ทำการตั้งชื่อ bucket จากนั้นทำการ Uncheck Block all public access

- ☐ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
- ☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant




Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.


☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

ทำการ check : I acknowledge that the current settings might result in this bucket and the objects within becoming public. ไม่ต้องแก้ไขส่วนที่เหลือ จากนั้นทำการ Create bucket

General purpose buckets (2) [Info](#) [All AWS Regions](#)

 [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)


Buckets are containers for data stored in S3.

< 1 > 

	Name ▲	AWS Region ▼	IAM Access Analyzer	Creation date ▼
<input type="radio"/>	thiencharoen3	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 14, 2024, 13:56:04 (UTC+07:00)

`จะเห็นได้ว่าการสร้าง bucket แล้ว

ให้กดเข้าไปใน bucket ที่เราเพิ่งสร้าง แล้วเข้าไปที่ Properties เพื่อทำ static website hosting โดยการเลือก Edit-> Enable ใน Index document ให้พิมพ์ไฟล์ .html ที่จะนำมาแสดง ในที่นี้ทางผู้จัดทำให้เป็น home.html แล้วกด Save changes


Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#) 


Static website hosting



☐ Disable

☒ Enable

Hosting type


☒ Host a static website
Use the bucket endpoint as the web address. [Learn more](#) 

☐ Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#) 

 For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#) 

Index document
Specify the home or default page of the website.

Error document - optional
This is returned when an error occurs.

Redirection rules - optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. [Learn more](#) 

กลับมาหน้า Objects แล้วทำการอัปโหลด folder สำหรับทำ hosting แล้วกดปุ่ม Upload

✔ Upload succeeded

View details below.

Summary

Destination

s3://thiencharoen-s3

Succeeded

✔ 34 files, 837.3 KB (100.00%)

Failed

⋮ 0 files, 0 B (0%)








Files and folders

Configuration

Files and folders (34 Total, 837.3 KB)

🔍 Find by name

< 1 2 3 4 >

Name	Folder ▾	Type ▾	Size ▾	Status ▾	Error ▾
background...	nm/img/	image/png	236.8 KB	✔ Succeeded	-
black.png 	nm/img/	image/png	1.5 KB	✔ Succeeded	-
cheese-Phot...	nm/img/	image/png	167.8 KB	✔ Succeeded	-
cheese.png 	nm/img/	image/png	9.3 KB	✔ Succeeded	-
gallery.png 	nm/img/	image/png	17.2 KB	✔ Succeeded	-
left-arrow.p... 	nm/img/	image/png	2.6 KB	✔ Succeeded	-
right-arrow... 	nm/img/	image/png	2.5 KB	✔ Succeeded	-
think-questi...	nm/img/	image/png	57.9 KB	✔ Succeeded	-
wishlist.png 	nm/img/	image/png	25.3 KB	✔ Succeeded	-
blue.jpg 	nm/	image/jpeg	7.6 KB	✔ Succeeded	-

เมื่อสำเร็จจะขึ้นตามภาพ แล้วกด close

จากนั้นเข้าไป permissions ที่อยู่ด้านข้าง Properties

thiencharoen-s3 Info

Objects

Properties

Permissions


Metrics

Management

Access Points

Permissions overview


Access finding

Access findings are provided by IAM external access analyzers. Learn more about [How IAM analyzer findings work](#) 

[View analyzer for ap-southeast-2](#)

Block public access (bucket settings)

Edit

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) 

Block all public access

 Off

► Individual Block Public Access settings for this bucket

แล้วทำการตั้งค่า Bucket policy โดยการกดปุ่ม Edit

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::thiencharoen-s3/*"
    }
  ]
}
```

ทำการพิมพ์ Policy ตามด้านบน แต่เปลี่ยน thiencharoen-s3 เป็นชื่อ bucket ของตัวเอง แล้วทำการ

Save changes

จากนั้นให้เรากลับไปไฟล์ที่เราทำการอัปโหลดทั้งหมดแล้วเลือกไปที่ไฟล์ html ที่เราได้ตั้งไว้เป็นหน้าแรกของ Web hosting เรา โดยทางผู้จัดทำจะเป็นไฟล์ home.html

Amazon S3 > Buckets > thiencharoen-s3 > nm/ > home.html

home.html Info Copy S3 URI Download Open Object actions ▼

Properties Permissions Versions

Object overview

Owner	chayapol.nem	S3 URI	s3://thiencharoen-s3/nm/home.html
AWS Region	Asia Pacific (Sydney) ap-southeast-2	Amazon Resource Name (ARN)	arn:aws:s3:::thiencharoen-s3/nm/home.html
Last modified	October 14, 2024, 14:13:33 (UTC+07:00)	Entity tag (Etag)	9dbf5ac117b5bb1410b7c3b8e5540f83
Size	1.3 KB	Object URL	https://thiencharoen-s3.s3.ap-southeast-2.amazonaws.com/nm/home.html
Type	html		
Key	nm/home.html		

Object URL จะเป็นที่อยู่ของเว็บไซต์ และโดเมนของ Web Host

โดยถ้าเรากดเข้าไปจะเป็นการเข้าไปที่เว็บไซต์ที่ได้ทำ Web Hosting แล้ว เป็นอันจบกระบวนการทำ Web Hosting

3. Link Demo การทำงาน <https://www.youtube.com/watch?v=greTr6GN9Zc>

4.อื่นๆ

4.1 ปัญหาและอุปสรรคที่เกิดขึ้นจากข้อจำกัดของ Software

ทรัพยากรมีจำกัดในด้านของ Cloud เช่น RAM ของ EC2 Free-Tier ไม่เพียงพอ ทำให้ต้องลดขนาด Model ลง การปรับ Model ให้มีประสิทธิภาพมากขึ้น ในขณะที่ขนาดของ Model ต้องลดลงหรือเท่าเดิม

4.2 ปัญหาและอุปสรรคที่เกิดขึ้นจากข้อจำกัดของ Hardware

การ Train AI จำเป็นต้องใช้ PC ของแต่ละบุคคล ซึ่งถ้าหากเป็น Model ขนาดใหญ่ หรือการ Implement ที่ไม่ดีพอ อาจทำให้ PC เกิดอาการค้างหรือกระตุกขึ้นได้

4.3 Future work

มีกระบวนการบอกชื่อ Cheese ที่มีข้อมูลตรงกับที่เราได้เลือกเพื่อเพิ่มเนื้อหาให้ครบถ้วนมากขึ้น ในส่วนของอัลกอริทึมรูปภาพเพื่อตรวจสอบว่าเป็นชีสประเภทอะไรจะมีการเพิ่มความแม่นยำมากขึ้น และมีการบอกรายละเอียดของชีสที่ได้จากการตรวจจับภาพมากขึ้น โดยเราจะเพิ่มทรัพยากรการประมวลผลของ Cloud ด้วย เพื่อเพิ่มฟังก์ชันที่หลากหลายในอนาคต