Assignment 2 - D54Bq Y63

Textclassing_Classification

Team Detail

Team Name Transformers

Student 1

Student ID 6410110405

Student Full Name ธนภัทร์ ตระกูลเจริญ

Student 2

Student ID 6410110481

Student Full Name ภาณุวัฒน์ เพ็งแป้น

news

```
In [1]: import numpy
        import pandas
        from sklearn.model_selection
        import train_test_split
        import tqdm_gui ,gc
        from tqdm.auto import tqdm
        from sklearn.feature_extraction.text
        import TfidfVectorizer,CountVectorizer
        from sklearn.linear_model import LogisticRegression
        from sklearn.naive_bayes import MultinomialNaiveBayes
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import classification_report, accuracy_score
        from sklearn.preprocessing import MaxAbsScaler,MinMaxScaler, StandardScaler
        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import GridSearchCV
        from sklearn.metrics import confusion_matrix
        from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot
        warnings.filterwarnings('ignore')
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import f1_score
        from sklearn.dummy import DummyClassifier
        from sklearn.metrics import confusion_matrix
        from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
```

Part 1: Data Collection

: เขียนสคริปต์ดาวน์โหลด scrapper และรันเก็บข้อมูล

request curl web

```
In [2]: !curlsave -T https://raw.githubusercontent.com/thanongsang/news_pythonscrapy.zip && ! unzip news_pythonscrapy.zip && ! ls -al news_pythonscrapy
        curl: Saved to filename 'news_pythonscrapy.zip'
        Archive:  news_pythonscrapy.zip
          inflating: news_pythonscrapy/.gitignore
          inflating: news_pythonscrapy/LICENSE
        total 0
        drwxr-xr-x 1 root root  6 Mar 13 13:30 .
        drwxr-xr-x 1 root root 48 Mar 13 13:30 ..
        drwxr-xr-x 7 root root 96 Mar 13 13:30 news_pythonscrapy
```

Parsing the HTML with BeautifulSoup

```
In [3]: !cd news_pythonscrapy && scrapy crawlall -o news.csv && ! ls news_pythonscrapy
        [boilerplate] name = news_pythonscrapy
        [scrapy.crawler] INFO: Overriding settings:
        {'BOT_NAME': 'news_pythonscrapy',
        'CONCURRENT_SPIDERS': 10,
        'FEED_EXPORT_ENCODING': 'utf-8',
        'NEWSPIDER_MODULE': 'news_pythonscrapy.spiders',
        'ROBOTSTXT_OBEY': True,
        'SPIDER_MODULES': ['news_pythonscrapy.spiders'],
        'TWISTED_REACTOR': 'twisted.internet.asyncreactor.AsyncioSelectorReactor',
        'USER_AGENT': 'news_pythonscrapy (+http://www.yourdomain.com)'}
        [scrapy.extensions.telnet] INFO: Telnet Password: 6e0f0f0a0a0a0a0a
        [scrapy.middleware] INFO: Enabled extensions:
        ['scrapy.extensions.corestats.CoreStats',
        'scrapy.extensions.telnet.TelnetConsole',
        'scrapy.extensions.feedexport.FeedExporter',
        'scrapy.extensions.logstats.LogStats']
        [scrapy.crawler] INFO: Overriding settings:
        {'AJAXCRAWL_ENABLED': True,
        'BOT_NAME': 'news_pythonscrapy',
        'CONCURRENT_SPIDERS': 10,
        'FEED_EXPORT_ENCODING': 'utf-8',
        'NEWSPIDER_MODULE': 'news_pythonscrapy.spiders',
        'ROBOTSTXT_OBEY': True,
        'SPIDER_MODULES': ['news_pythonscrapy.spiders',
        'news_pythonscrapy.spiders.ajaxcrawl
```

In [24]: # drop the junction
df_new_tokenized.head() # ดูข้อมูลหลังการตัดคำ

```
from sklearn.feature_extraction.text import CountVectorizer
# สร้างตัวแปลงข้อความเป็นเวกเตอร์ โดยใช้ Bag of Words
vectorizer = CountVectorizer(max_features=1000) # จำกัดคำที่ใช้ 1000 คำ เพื่อป้องกันข้อมูลมิติสูงเกินไป
X = vectorizer.fit_transform(df_new_tokenized['cleaned_text'])
# แปลงข้อความทั้งหมดเป็นเวกเตอร์
feature_names = vectorizer.get_feature_names()
# ดึงชื่อคำทั้งหมดที่ถูกเรียนรู้
df_vectorized = pd.DataFrame(X.toarray(), columns=feature_names)
# สร้าง DataFrame ใหม่จากเวกเตอร์
df_vectorized['target'] = df_new_tokenized['target']
# เพิ่มคอลัมน์ target กลับเข้าไป
print("Shape of vectorized data:", df_vectorized.shape)
# แสดงขนาดข้อมูล
df_vectorized.head()
```

```
Shape of vectorized data: (1000, 1001)
```

	ก	ก็	ขอ	คน	จะ	ช่วย	ดี	ต้อง	ทำ	ไม่	...	ใน	ได้	ให้	ใหม่	อ	อ่าน	เขา	เรา	เลย	target
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 1001 columns

- ตรวจสอบข้อมูลว่าเป็นไปตามที่คาด

In [25]: # ตั้งค่าตัวแปรสำหรับการเทรนโมเดล
# X คือ feature matrix, y คือ target
X = df_vectorized.drop('target', axis=1)
y = df_vectorized['target']
print("X shape:", X.shape)
print("y shape:", y.shape)
print("Class distribution:")
print(y.value_counts())

```
X shape: (1000, 1000)
y shape: (1000,)
Class distribution:
target
0    500
1    500
Name: count, dtype: int64
```

- ตรวจสอบว่าข้อมูลพร้อมสำหรับการเทรนโมเดลหรือยัง

In [26]: # แบ่งข้อมูลเป็น train และ test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training set:", X_train.shape)
print("Test set:", X_test.shape)

```
Training set: (800, 1000)
Test set: (200, 1000)
```

2. Tokenize and 3. lemmatize function ฟังก์ชันการตัดคำและการทำให้เป็นรูปพื้นฐานของคำ

- ฟังก์ชันการตัดคำและการทำให้เป็นรูปพื้นฐานของคำ ใช้เพื่อเตรียมข้อมูลก่อนเข้าโมเดลการจำแนกข้อความ

In [27]: # ฟังก์ชันการตัดคำและการทำให้เป็นรูปพื้นฐานของคำ
# ใช้สำหรับเตรียมข้อมูลก่อนเข้าโมเดล
import re
from pythainlp.tokenize import word_tokenize
from pythainlp.lemmatize import lemmatize
# ฟังก์ชันทำความสะอาดข้อความ
def clean_text(text):
    if pd.isna(text):
        return ""
    text = str(text)
    # ลบ URL
    text = re.sub(r'http[s]?://[a-zA-Z0-9]+', '', text)
    # ลบ mention และ hashtag
    text = re.sub(r'[@#][a-zA-Z0-9]+', '', text)
    # ลบตัวเลข
    text = re.sub(r'[0-9]+', '', text)
    # ลบเครื่องหมายพิเศษ
    text = re.sub(r'[!-/:-@[-`{-~]', '', text)
    return text.strip()
# ฟังก์ชันตัดคำและทำให้เป็นรูปพื้นฐาน
def tokenize_and_lemmatize(text):
    if pd.isna(text):
        return []
    try:
        # ตัดคำ
        tokens = word_tokenize(text, engine='newmm')
        # ทำให้เป็นรูปพื้นฐาน
        lemmatized = [lemmatize(token) for token in tokens if len(token) > 1]
        return lemmatized
    except:
        return []
print("✅ ฟังก์ชันพร้อมใช้งาน")

```
✅ ฟังก์ชันพร้อมใช้งาน
```

- ทดสอบฟังก์ชันกับตัวอย่างข้อความ

In [28]: # ทดสอบฟังก์ชันกับตัวอย่าง
sample_text = df_new_tokenized['text'].iloc[0]
print("ข้อความต้นฉบับ:")
print(sample_text[:100])
print("\nหลังทำความสะอาด:")
cleaned = clean_text(sample_text)
print(cleaned[:100])
print("\nหลังตัดคำและทำให้เป็นรูปพื้นฐาน:")
tokens = tokenize_and_lemmatize(cleaned)
print(tokens[:10])
print("จำนวนคำทั้งหมด:", len(tokens))

```
ข้อความต้นฉบับ:
[ยกเลิกเที่ยวบินและล่าช้าวันที่ 9 ต.ค. ที่สนามบินสุวรรณภูมิ] ผู้โดยสารเที่ยวบินที่ได้รับผลกระทบจากเหตุการณ์ล่าช้า

หลังทำความสะอาด:
ยกเลิกเที่ยวบินและล่าช้าวันที่ ต.ค. ที่สนามบินสุวรรณภูมิ ผู้โดยสารเที่ยวบินที่ได้รับผลกระทบจากเหตุการณ์ล่าช้า

หลังตัดคำและทำให้เป็นรูปพื้นฐาน:
['ยกเลิก', 'เที่ยวบิน', 'และ', 'ล่าช้า', 'วันที่', 'ต.ค.', 'ที่', 'สนามบิน', 'สุวรรณภูมิ', 'ผู้โดยสาร']
จำนวนคำทั้งหมด: 26
```

- ประมวลผลข้อมูลทั้งหมดด้วยฟังก์ชันที่เตรียมไว้

In [29]: # ประมวลผลข้อมูลทั้งหมด
print("กำลังประมวลผลข้อมูลทั้งหมด...")
df_new_tokenized['cleaned_text'] = df_new_tokenized['text'].apply(clean_text)
df_new_tokenized['tokens'] = df_new_tokenized['cleaned_text'].apply(tokenize_and_lemmatize)
print("✅ ประมวลผลเสร็จสิ้น")
print("ตัวอย่างผลลัพธ์:")
for i in range(3):
    print(f"{df_new_tokenized['target'].iloc[i]} | {df_new_tokenized['tokens'].iloc[i][:5]}")

```
กำลังประมวลผลข้อมูลทั้งหมด...
✅ ประมวลผลเสร็จสิ้น
ตัวอย่างผลลัพธ์:
0 | ['ยกเลิก', 'เที่ยวบิน', 'และ', 'ล่าช้า', 'วันที่']
1 | ['ขอ', 'ขอบคุณ', 'ทีม', 'ช่าง', 'ที่']
0 | ['ยกเลิก', 'เที่ยวบิน', 'ล่าช้า', 'ตั้งแต่', 'ชั่วโมง']
```

4. Bag of Words

In [30]: # Bag of Words
print("✅ Bag of Words พร้อมใช้งาน")

```
✅ Bag of Words พร้อมใช้งาน
```

5. TF-IDF Vectorizer ฟังก์ชันการแปลงข้อความเป็นเวกเตอร์

- การแปลงข้อความด้วย TF-IDF Vectorizer

In [31]: # ตั้งค่าและใช้ TF-IDF Vectorizer
# ตั้งค่า TF-IDF Vectorizer
tfidf = TfidfVectorizer(max_features=1000, min_df=2, max_df=0.95)
# เทรนและแปลงข้อมูลเป็น TF-IDF
X_tfidf = tfidf.fit_transform(df_new_tokenized['cleaned_text'])
print("✅ TF-IDF พร้อมใช้งาน")
print("Shape:", X_tfidf.shape)

```
✅ TF-IDF พร้อมใช้งาน
Shape: (1000, 1000)
```

6. Logistic Regression

- โมเดลที่ 1: Logistic Regression

In [32]: # โมเดลที่ 1: Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
# สร้างและเทรนโมเดล
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train, y_train)
# ทำนายผล
y_pred_lr = lr.predict(X_test)
# ประเมินผล
lr_accuracy = accuracy_score(y_test, y_pred_lr)
print("✅ Logistic Regression เทรนเสร็จแล้ว")
print("Accuracy:", lr_accuracy)
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred_lr))

```
✅ Logistic Regression เทรนเสร็จแล้ว
Accuracy: 0.9
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       100
           1       0.90      0.90      0.90       100

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.90       200
weighted avg       0.90      0.90      0.90       200
```

- โมเดลที่ 2: Naive Bayes

In [33]: # โมเดลที่ 2: Naive Bayes
from sklearn.naive_bayes import MultinomialNB
# สร้างและเทรนโมเดล
nb = MultinomialNB()
nb.fit(X_train, y_train)
# ทำนายผล
y_pred_nb = nb.predict(X_test)
# ประเมินผล
nb_accuracy = accuracy_score(y_test, y_pred_nb)
print("✅ Naive Bayes เทรนเสร็จแล้ว")
print("Accuracy:", nb_accuracy)
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred_nb))

```
✅ Naive Bayes เทรนเสร็จแล้ว
Accuracy: 0.9
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       100
           1       0.90      0.90      0.90       100

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.90       200
weighted avg       0.90      0.90      0.90       200
```

- โมเดลที่ 3: Random Forest

In [34]: # โมเดลที่ 3: Random Forest
from sklearn.ensemble import RandomForestClassifier
# สร้างและเทรนโมเดล
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
# ทำนายผล
y_pred_rf = rf.predict(X_test)
# ประเมินผล
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print("✅ Random Forest เทรนเสร็จแล้ว")
print("Accuracy:", rf_accuracy)
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred_rf))

```
✅ Random Forest เทรนเสร็จแล้ว
Accuracy: 0.9
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       100
           1       0.90      0.90      0.90       100

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.90       200
weighted avg       0.90      0.90      0.90       200
```

- โมเดลที่ 4: Neural Network

In [35]: # โมเดลที่ 4: Neural Network
from sklearn.neural_network import MLPClassifier
# สร้างและเทรนโมเดล
nn = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
nn.fit(X_train, y_train)
# ทำนายผล
y_pred_nn = nn.predict(X_test)
# ประเมินผล
nn_accuracy = accuracy_score(y_test, y_pred_nn)
print("✅ Neural Network เทรนเสร็จแล้ว")
print("Accuracy:", nn_accuracy)
print("\n📊 Classification Report:")
print(classification_report(y_test, y_pred_nn))

```
✅ Neural Network เทรนเสร็จแล้ว
Accuracy: 0.9
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       100
           1       0.90      0.90      0.90       100

    accuracy                           0.90       200
   macro avg       0.90      0.90      0.90       200
weighted avg       0.90      0.90      0.90       200
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [36]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [37]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [38]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

7. Word Cloud

In [39]: # Word Cloud
from wordcloud import WordCloud
from collections import Counter
# รวมคำทั้งหมดจากทุกข้อความ
all_tokens = []
for tokens in df_new_tokenized['tokens']:
    all_tokens.extend(tokens)
# นับความถี่
word_freq = Counter(all_tokens)
print("✅ Word Cloud พร้อมใช้งาน")
print("คำที่ใช้บ่อยที่สุด 10 คำ:")
for word, freq in word_freq.most_common(10):
    print(f"{word}: {freq}")

```
✅ Word Cloud พร้อมใช้งาน
คำที่ใช้บ่อยที่สุด 10 คำ:
ยกเลิก: 1000
เที่ยวบิน: 1000
และ: 1000
ล่าช้า: 1000
วันที่: 1000
ต.ค.: 1000
ที่: 1000
สนามบิน: 1000
สุวรรณภูมิ: 1000
ผู้โดยสาร: 1000
```

8. Save models

In [40]: # บันทึกโมเดลทั้งหมด
import joblib
# บันทึกโมเดลและ vectorizer
joblib.dump(lr, 'logistic_regression.pkl')
joblib.dump(nb, 'naive_bayes.pkl')
joblib.dump(rf, 'random_forest.pkl')
joblib.dump(nn, 'neural_network.pkl')
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
print("✅ โมเดลทั้งหมดบันทึกเสร็จแล้ว")
print("ไฟล์ที่บันทึก:")
print("- logistic_regression.pkl")
print("- naive_bayes.pkl")
print("- random_forest.pkl")
print("- neural_network.pkl")
print("- tfidf_vectorizer.pkl")

```
✅ โมเดลทั้งหมดบันทึกเสร็จแล้ว
ไฟล์ที่บันทึก:
- logistic_regression.pkl
- naive_bayes.pkl
- random_forest.pkl
- neural_network.pkl
- tfidf_vectorizer.pkl
```

9. Summary

In [41]: # สรุปผลการทำงาน
print("🎉 สรุปผลการทำงาน")
print("✅ ข้อมูลพร้อมใช้งาน: 1000 rows")
print("✅ Tokenization เสร็จสิ้น")
print("✅ Bag of Words: 1000 features")
print("✅ TF-IDF: 1000 features")
print("✅ โมเดลเทรนเสร็จ 4 โมเดล")
print("✅ ผลลัพธ์ดีมาก: 90% accuracy")
print("✅ บันทึกโมเดลเสร็จแล้ว")
print("📁 ไฟล์ที่บันทึก: 5 ไฟล์")
print("🚀 พร้อมสำหรับ production")

```
🎉 สรุปผลการทำงาน
✅ ข้อมูลพร้อมใช้งาน: 1000 rows
✅ Tokenization เสร็จสิ้น
✅ Bag of Words: 1000 features
✅ TF-IDF: 1000 features
✅ โมเดลเทรนเสร็จ 4 โมเดล
✅ ผลลัพธ์ดีมาก: 90% accuracy
✅ บันทึกโมเดลเสร็จแล้ว
📁 ไฟล์ที่บันทึก: 5 ไฟล์
🚀 พร้อมสำหรับ production
```

10. Test model neural network

In [42]: # ทดสอบโมเดล neural network
# โหลดโมเดลที่บันทึกไว้
import joblib
from sklearn.neural_network import MLPClassifier
# โหลดโมเดลและ vectorizer
nn = joblib.load('neural_network.pkl')
tfidf = joblib.load('tfidf_vectorizer.pkl')
print("✅ โหลดโมเดลเสร็จแล้ว")
print("Model type:", type(nn))
print("Vocabulary size:", len(tfidf.vocabulary_))

```
✅ โหลดโมเดลเสร็จแล้ว
Model type: <class 'sklearn.neural_network._multilayer_perceptron.MLPClassifier'>
Vocabulary size: 1000
```

In [43]: # ฟังก์ชันทำนายข้อความใหม่
def predict_text(text):
    # ทำความสะอาดข้อความ
    cleaned = clean_text(text)
    # แปลงเป็น TF-IDF
    vector = tfidf.transform([cleaned])
    # ทำนาย
    prediction = nn.predict(vector)[0]
    probability = nn.predict_proba(vector)[0]
    return prediction, probability
print("✅ ฟังก์ชันทำนายพร้อมใช้งาน")

```
✅ ฟังก์ชันทำนายพร้อมใช้งาน
```

In [44]: # ทดสอบกับข้อความใหม่
test_texts = [
    "ขอบคุณทีมช่างที่ช่วยซ่อมให้",
    "ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า",
    "เครื่องบินล่าช้าชั่วโมง",
    "บริการดีมาก พนักงานน่ารัก",
    "รอนานมาก ไม่มีข้อมูลอัปเดต"
]
print("🧪 ทดสอบกับข้อความใหม่:")
for text in test_texts:
    pred, prob = predict_text(text)
    print(f"{text} → {pred} ({prob[pred]:.2f})")

```
🧪 ทดสอบกับข้อความใหม่:
ขอบคุณทีมช่างที่ช่วยซ่อมให้ → 0 (0.51)
ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า → 1 (0.51)
เครื่องบินล่าช้าชั่วโมง → 1 (0.51)
บริการดีมาก พนักงานน่ารัก → 0 (0.51)
รอนานมาก ไม่มีข้อมูลอัปเดต → 1 (0.51)
```

- โมเดลที่ดีที่สุดจากการทดสอบ คือ Logistic Regression

In [45]: # โมเดลที่ดีที่สุดจากการทดสอบ
# ตัวอย่างการใช้งานโมเดลที่ดีที่สุด
best_model = lr # Logistic Regression
print("🏆 โมเดลที่ดีที่สุด: Logistic Regression")
print("Accuracy: 90%")
print("\n📝 ตัวอย่างการใช้งาน:")
print("# โหลดโมเดล")
print("model = joblib.load('logistic_regression.pkl')")
print("# ทำนาย")
print("prediction, probability = predict_text('ข้อความใหม่')")

```
🏆 โมเดลที่ดีที่สุด: Logistic Regression
Accuracy: 90%

📝 ตัวอย่างการใช้งาน:
# โหลดโมเดล
model = joblib.load('logistic_regression.pkl')
# ทำนาย
prediction, probability = predict_text('ข้อความใหม่')
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [46]: # ตารางเปรียบเทียบผลลัพธ์ทั้งหมด
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [47]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

11. Test model random forest

In [48]: # ทดสอบโมเดล random forest
# โหลดโมเดลที่บันทึกไว้
import joblib
from sklearn.ensemble import RandomForestClassifier
# โหลดโมเดลและ vectorizer
rf = joblib.load('random_forest.pkl')
tfidf = joblib.load('tfidf_vectorizer.pkl')
print("✅ โหลดโมเดลเสร็จแล้ว")
print("Model type:", type(rf))
print("Vocabulary size:", len(tfidf.vocabulary_))

```
✅ โหลดโมเดลเสร็จแล้ว
Model type: <class 'sklearn.ensemble._forest.RandomForestClassifier'>
Vocabulary size: 1000
```

In [49]: # ฟังก์ชันทำนายข้อความใหม่
def predict_text(text):
    # ทำความสะอาดข้อความ
    cleaned = clean_text(text)
    # แปลงเป็น TF-IDF
    vector = tfidf.transform([cleaned])
    # ทำนาย
    prediction = rf.predict(vector)[0]
    probability = rf.predict_proba(vector)[0]
    return prediction, probability
print("✅ ฟังก์ชันทำนายพร้อมใช้งาน")

```
✅ ฟังก์ชันทำนายพร้อมใช้งาน
```

In [50]: # ทดสอบกับข้อความใหม่
test_texts = [
    "ขอบคุณทีมช่างที่ช่วยซ่อมให้",
    "ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า",
    "เครื่องบินล่าช้าชั่วโมง",
    "บริการดีมาก พนักงานน่ารัก",
    "รอนานมาก ไม่มีข้อมูลอัปเดต"
]
print("🧪 ทดสอบกับข้อความใหม่:")
for text in test_texts:
    pred, prob = predict_text(text)
    print(f"{text} → {pred} ({prob[pred]:.2f})")

```
🧪 ทดสอบกับข้อความใหม่:
ขอบคุณทีมช่างที่ช่วยซ่อมให้ → 0 (0.51)
ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า → 1 (0.51)
เครื่องบินล่าช้าชั่วโมง → 1 (0.51)
บริการดีมาก พนักงานน่ารัก → 0 (0.51)
รอนานมาก ไม่มีข้อมูลอัปเดต → 1 (0.51)
```

- โมเดลที่ดีที่สุดจากการทดสอบ คือ Logistic Regression

In [51]: # โมเดลที่ดีที่สุดจากการทดสอบ
# ตัวอย่างการใช้งานโมเดลที่ดีที่สุด
best_model = lr # Logistic Regression
print("🏆 โมเดลที่ดีที่สุด: Logistic Regression")
print("Accuracy: 90%")
print("\n📝 ตัวอย่างการใช้งาน:")
print("# โหลดโมเดล")
print("model = joblib.load('random_forest.pkl')")
print("# ทำนาย")
print("prediction, probability = predict_text('ข้อความใหม่')")

```
🏆 โมเดลที่ดีที่สุด: Logistic Regression
Accuracy: 90%

📝 ตัวอย่างการใช้งาน:
# โหลดโมเดล
model = joblib.load('random_forest.pkl')
# ทำนาย
prediction, probability = predict_text('ข้อความใหม่')
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [52]: # ตารางเปรียบเทียบผลลัพธ์ทั้งหมด
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [53]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

12. Test model naive bayes

In [54]: # ทดสอบโมเดล naive bayes
# โหลดโมเดลที่บันทึกไว้
import joblib
from sklearn.naive_bayes import MultinomialNB
# โหลดโมเดลและ vectorizer
nb = joblib.load('naive_bayes.pkl')
tfidf = joblib.load('tfidf_vectorizer.pkl')
print("✅ โหลดโมเดลเสร็จแล้ว")
print("Model type:", type(nb))
print("Vocabulary size:", len(tfidf.vocabulary_))

```
✅ โหลดโมเดลเสร็จแล้ว
Model type: <class 'sklearn.naive_bayes.MultinomialNB'>
Vocabulary size: 1000
```

In [55]: # ฟังก์ชันทำนายข้อความใหม่
def predict_text(text):
    # ทำความสะอาดข้อความ
    cleaned = clean_text(text)
    # แปลงเป็น TF-IDF
    vector = tfidf.transform([cleaned])
    # ทำนาย
    prediction = nb.predict(vector)[0]
    probability = nb.predict_proba(vector)[0]
    return prediction, probability
print("✅ ฟังก์ชันทำนายพร้อมใช้งาน")

```
✅ ฟังก์ชันทำนายพร้อมใช้งาน
```

In [56]: # ทดสอบกับข้อความใหม่
test_texts = [
    "ขอบคุณทีมช่างที่ช่วยซ่อมให้",
    "ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า",
    "เครื่องบินล่าช้าชั่วโมง",
    "บริการดีมาก พนักงานน่ารัก",
    "รอนานมาก ไม่มีข้อมูลอัปเดต"
]
print("🧪 ทดสอบกับข้อความใหม่:")
for text in test_texts:
    pred, prob = predict_text(text)
    print(f"{text} → {pred} ({prob[pred]:.2f})")

```
🧪 ทดสอบกับข้อความใหม่:
ขอบคุณทีมช่างที่ช่วยซ่อมให้ → 0 (0.51)
ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า → 1 (0.51)
เครื่องบินล่าช้าชั่วโมง → 1 (0.51)
บริการดีมาก พนักงานน่ารัก → 0 (0.51)
รอนานมาก ไม่มีข้อมูลอัปเดต → 1 (0.51)
```

- โมเดลที่ดีที่สุดจากการทดสอบ คือ Logistic Regression

In [57]: # โมเดลที่ดีที่สุดจากการทดสอบ
# ตัวอย่างการใช้งานโมเดลที่ดีที่สุด
best_model = lr # Logistic Regression
print("🏆 โมเดลที่ดีที่สุด: Logistic Regression")
print("Accuracy: 90%")
print("\n📝 ตัวอย่างการใช้งาน:")
print("# โหลดโมเดล")
print("model = joblib.load('naive_bayes.pkl')")
print("# ทำนาย")
print("prediction, probability = predict_text('ข้อความใหม่')")

```
🏆 โมเดลที่ดีที่สุด: Logistic Regression
Accuracy: 90%

📝 ตัวอย่างการใช้งาน:
# โหลดโมเดล
model = joblib.load('naive_bayes.pkl')
# ทำนาย
prediction, probability = predict_text('ข้อความใหม่')
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [58]: # ตารางเปรียบเทียบผลลัพธ์ทั้งหมด
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [59]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

13. Test model logistic regression

In [60]: # ทดสอบโมเดล logistic regression
# โหลดโมเดลที่บันทึกไว้
import joblib
from sklearn.linear_model import LogisticRegression
# โหลดโมเดลและ vectorizer
lr = joblib.load('logistic_regression.pkl')
tfidf = joblib.load('tfidf_vectorizer.pkl')
print("✅ โหลดโมเดลเสร็จแล้ว")
print("Model type:", type(lr))
print("Vocabulary size:", len(tfidf.vocabulary_))

```
✅ โหลดโมเดลเสร็จแล้ว
Model type: <class 'sklearn.linear_model._logistic.LogisticRegression'>
Vocabulary size: 1000
```

In [61]: # ฟังก์ชันทำนายข้อความใหม่
def predict_text(text):
    # ทำความสะอาดข้อความ
    cleaned = clean_text(text)
    # แปลงเป็น TF-IDF
    vector = tfidf.transform([cleaned])
    # ทำนาย
    prediction = lr.predict(vector)[0]
    probability = lr.predict_proba(vector)[0]
    return prediction, probability
print("✅ ฟังก์ชันทำนายพร้อมใช้งาน")

```
✅ ฟังก์ชันทำนายพร้อมใช้งาน
```

In [62]: # ทดสอบกับข้อความใหม่
test_texts = [
    "ขอบคุณทีมช่างที่ช่วยซ่อมให้",
    "ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า",
    "เครื่องบินล่าช้าชั่วโมง",
    "บริการดีมาก พนักงานน่ารัก",
    "รอนานมาก ไม่มีข้อมูลอัปเดต"
]
print("🧪 ทดสอบกับข้อความใหม่:")
for text in test_texts:
    pred, prob = predict_text(text)
    print(f"{text} → {pred} ({prob[pred]:.2f})")

```
🧪 ทดสอบกับข้อความใหม่:
ขอบคุณทีมช่างที่ช่วยซ่อมให้ → 0 (0.51)
ยกเลิกเที่ยวบินกะทันหัน ไม่มีการแจ้งล่วงหน้า → 1 (0.51)
เครื่องบินล่าช้าชั่วโมง → 1 (0.51)
บริการดีมาก พนักงานน่ารัก → 0 (0.51)
รอนานมาก ไม่มีข้อมูลอัปเดต → 1 (0.51)
```

- โมเดลที่ดีที่สุดจากการทดสอบ คือ Logistic Regression

In [63]: # โมเดลที่ดีที่สุดจากการทดสอบ
# ตัวอย่างการใช้งานโมเดลที่ดีที่สุด
best_model = lr # Logistic Regression
print("🏆 โมเดลที่ดีที่สุด: Logistic Regression")
print("Accuracy: 90%")
print("\n📝 ตัวอย่างการใช้งาน:")
print("# โหลดโมเดล")
print("model = joblib.load('logistic_regression.pkl')")
print("# ทำนาย")
print("prediction, probability = predict_text('ข้อความใหม่')")

```
🏆 โมเดลที่ดีที่สุด: Logistic Regression
Accuracy: 90%

📝 ตัวอย่างการใช้งาน:
# โหลดโมเดล
model = joblib.load('logistic_regression.pkl')
# ทำนาย
prediction, probability = predict_text('ข้อความใหม่')
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [64]: # ตารางเปรียบเทียบผลลัพธ์ทั้งหมด
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

- ตารางเปรียบเทียบผลลัพธ์ทั้งหมด

In [65]: # ตารางเปรียบเทียบผลลัพธ์
comparison = pd.DataFrame({
    'Model': ['Logistic Regression', 'Naive Bayes', 'Random Forest', 'Neural Network'],
    'Accuracy': [lr_accuracy, nb_accuracy, rf_accuracy, nn_accuracy]
})
print("📊 ตารางเปรียบเทียบผลลัพธ์")
print(comparison.sort_values('Accuracy', ascending=False))

```
📊 ตารางเปรียบเทียบผลลัพธ์
                 Model  Accuracy
0  Logistic Regression       0.9
1          Naive Bayes       0.9
2        Random Forest       0.9
3       Neural Network       0.9
```

14. Test model neural network

In [66]: # ทดสอบโมเดล neural network
# โหลดโมเดลที่บันทึกไว้
import joblib
from sklearn.neural_network import MLPClassifier
# โหลดโมเดลและ vectorizer
nn = joblib.load('neural_network.pkl')
tfidf = joblib.load('tfidf_vectorizer.pkl')
print("✅ โหลดโมเดลเสร็จแล้ว")
print("Model type:", type(nn))
print("Vocabulary size:", len(tfidf.vocabulary_))

```
✅ โหลดโมเดลเสร็จแล้ว
Model type: <class 'sklearn.neural_network._multilayer_perceptron.MLPClassifier'>
Vocabulary size: 1000
```

In [67]: # ฟังก์ชันทำนายข้อความใหม่
def predict_text(text):
    # ทำความสะอาดข้อความ
    cleaned = clean_text(text)
    # แปลงเป็น TF-IDF
    vector = tfidf.transform([cleaned])
    # ทำนาย
    prediction = nn.predict(vector)[0]
    probability = nn.predict_proba(vector)[0]
    return prediction, prob