This is the edited version of report. Only included the methodology and result part.

1.2. วัตถุประสงค์ของโครงงาน

- พัฒนาโมเดลที่สามารถจำแนกเพศของผู้ใช้งานโซเชียลมีเดียจากชื่อผู้ใช้งาน (username) โดยใช้ machine learning ซึ่งจะต้องมีการเก็บข้อมูลเป็นอย่างมากเพื่อที่จะสามารถจำแนกเพศของผู้ใช้งานจากชื่อผู้ใช้งานได้อย่างแม่นยำที่สูง

- พัฒนาโมเดลที่สามารถจำแนกเพศของผู้ใช้งานโดยทำการทดสอบหลายๆโมเดลเพื่อให้ได้ผลการทดสอบการจำแนกเพศมีความแม่นยำที่สูง

- พัฒนาโมเดลที่สามารถจำแนกเพศของผู้ใช้งานโดยทำการเพิ่ม feature ต่างๆให้โมเดล เพื่อให้ได้ผลการทดสอบการจำแนกเพศมีความแม่นยำที่สูง

1.3. ขอบเขตของโครงงาน

- โครงงานนี้จะสนใจแค่เพียงชื่อภาษาอังกฤษเท่านั้น เนื่องจากทำการต่อยอดจากโครงงานที่ทำการจำแนกเพศด้วยชื่อในภาษาไทยเท่านั้น

- ในส่วนของการจำแนกเพศของผู้ใช้งานโซเชียลมีเดียจากชื่อจริงจะเป็นการดึงข้อมูลผู้ใช้งานมาจาก facebook เท่านั้น

1.5. ประโยชน์ที่คาดว่าจะได้รับ

เมื่อเราสามารถจำแนกเพศของผู้ใช้งานโซเชียลมีเดียได้ เราก็สามารถดึงข้อมูลต่างๆทางโซเชียลมีเดียมาวิเคราะห์ได้ เช่น โพสต์ๆหนึ่งใน facebook เราก็สามารถนำโพสต์นั้นมาวิเคราะห์ข้อมูลว่าผู้ที่สนใจในโพสต์ๆนั้นตรงกับความต้องการของเจ้าของโพสหรือไม่ ซึ่งถ้าโพสต์ๆนั้นไม่ตรงตามความต้องการของเจ้าของโพสก็อาจจะต้องทำการปรับเปลี่ยนวิธีการโพสเพื่อให้ผู้ที่เข้าถึงโพสต์นั้นตรงตามความต้องการ

3. Methodology

3.1. Dataset

เราได้ทำการเก็บข้อมูล Username คนไทยในภาษาอังกฤษจาก Facebook ด้วยการเก็บด้วยตนเองตั้งแต่เดือนพฤศจิกายน 2562 – เดือนพฤษภาคม 2563 เราได้เลือกเฉพาะผู้ใช้งานคนไทยที่มี username เป็นภาษาอังกฤษและมีเพศระบุไว้ใน profile เท่านั้น ใน dataset จะประกอบไปด้วย 10,031 รายชื่อซึ่งเป็นเพศชายจำนวน 4571 ชื่อ(45.57%)และเพศหญิงจำนวน 5460 ชื่อ(54.43%) ตัวอย่าง dataset ที่เราเก็บมาเป็นดังตารางที่ 1

Username	Gender
Yuwadee Klanarong Civil	Female
Muay Chita	Female
Passorn DT	Female
Nut Samsarai	Male
Jan Jao	Female
Janista Sumranthin	Female
Rungtip Piao	Female
Bank Pathompong	Male
Satit Ann	Male
Mayla Kewalin	Female

ตารางที่ 1 ตัวอย่างตาราง dataset

3.2. Character frequency feature

คือ การแยกนับแต่ละตัวอักษรจากคำนั้นๆ สำหรับชื่อจริงภาษาอังกฤษจะประกอบด้วยตัวอักษร 26 ตัว ตั้งแต่ a-z แต่สำหรับชื่อผู้ใช้งานจะประกอบไปด้วยตัวอักษร a-z ตัวเลข และอักขระพิเศษ เช่น . , - , ’ เป็นต้น ดังแสดงในตารางที่ 2

Firstname	Frequency character
Yuwadee	{'y': 1, 'u': 1, 'w': 1, 'a': 1, 'd': 1, 'e': 2}
Muay	{'m': 1, 'u': 1, 'a': 1, 'y': 1}
Passorn	{'p': 1, 'a': 1, 's': 2, 'o': 1, 'r': 1, 'n': 1}
Nut	{'n': 1, 'u': 1, 't': 1}
Jan	{'j': 1, 'a': 1, 'n': 1}
Janista	{'j': 1, 'a': 2, 'n': 1, 'i': 1, 's': 1, 't': 1}
Rungtip	{'r': 1, 'u': 1, 'n': 1, 'g': 1, 't': 1, 'i': 1, 'p': 1}
Bank	{'b': 1, 'a': 1, 'n': 1, 'k': 1}
Satit	{'s': 1, 'a': 1, 't': 2, 'i': 1}
Mayla	{'m': 1, 'a': 2, 'y': 1, 'l': 1}

ตารางที่ 2 ตัวอย่างตารางความถี่ตัวอักษร

3.3. Substring character feature

คือ กลุ่มของตัวอักษรจากคำนั้นๆ การแบ่งของ substring สามารถแบ่งได้หลายแบบ เช่น แบบ 2 ตัวอักษร แบบ 3 ตัวอักษร แบบ 3 ตัวอักษรแรก แบบ 3 ตัวอักษรหลัง เป็นต้น ดังแสดงในตารางที่ 3

Firstname	Substring character
Yuwadee	{'first_letter': 'y', 'first2_letter': 'yu', 'first3_letter': 'yuw', 'first4_letter': 'yuwa', 'last_letter': 'e', 'last2_letter': 'ee', 'last3_letter': 'dee', 'last4_letter': 'adee'}
Muay	{'first_letter': 'm', 'first2_letter': 'mu', 'first3_letter': 'mua', 'first4_letter': 'muay', 'last_letter': 'y', 'last2_letter': 'ay', 'last3_letter': 'uay', 'last4_letter': 'muay'}
Passorn	{'first_letter': 'p', 'first2_letter': 'pa', 'first3_letter': 'pas', 'first4_letter': 'pass', 'last_letter': 'n', 'last2_letter': 'rn', 'last3_letter': 'orn', 'last4_letter': 'sorn'}
Nut	{'first_letter': 'n', 'first2_letter': 'nu', 'first3_letter': 'nut', 'first4_letter': 'nut', 'last_letter': 't', 'last2_letter': 'ut', 'last3_letter': 'nut', 'last4_letter': 'nut'}
Jan	{'first_letter': 'j', 'first2_letter': 'ja', 'first3_letter': 'jan', 'first4_letter': 'jan', 'last_letter': 'n', 'last2_letter': 'an', 'last3_letter': 'jan', 'last4_letter': 'jan'}
Janista	{'first_letter': 'j', 'first2_letter': 'ja', 'first3_letter': 'jan', 'first4_letter': 'jani', 'last_letter': 'a', 'last2_letter': 'ta', 'last3_letter': 'sta', 'last4_letter': 'ista'}

Rungtip	{'first_letter': 'r', 'first2_letter': 'ru', 'first3_letter': 'run', 'first4_letter': 'rung', 'last_letter': 'p', 'last2_letter': 'ip', 'last3_letter': 'tip', 'last4_letter': 'gtip'}
Bank	{'first_letter': 'b', 'first2_letter': 'ba', 'first3_letter': 'ban', 'first4_letter': 'bank', 'last_letter': 'k', 'last2_letter': 'nk', 'last3_letter': 'ank', 'last4_letter': 'bank'}
Satit	{'first_letter': 's', 'first2_letter': 'sa', 'first3_letter': 'sat', 'first4_letter': 'sati', 'last_letter': 't', 'last2_letter': 'it', 'last3_letter': 'tit', 'last4_letter': 'atit'}
Mayla	{'first_letter': 'm', 'first2_letter': 'ma', 'first3_letter': 'may', 'first4_letter': 'mayl', 'last_letter': 'a', 'last2_letter': 'la', 'last3_letter': 'yla', 'last4_letter': 'ayla'}

ตารางที่ 3 ตัวอย่างตารางกลุ่มของตัวอักษร

3.4 Syllable N-grams

คือ ลำดับของพยางค์ที่ติดกันจำนวน n โดยจำนวน n สามารถเป็นจำนวนใดก็ได้แล้วแต่กำหนด แต่ในรายงานนี้จะให้เป็น 1-2 เท่านั้นเพราะว่าจำนวนพยางค์ชื่อของคนไทยส่วนใหญ่จะอยู่ที่ 2-3 พยางค์เพราะว่าในกรณีที่จำนวน n มากกว่าจำนวนพยางค์จะไม่มีลำดับของชื่อนั้นๆ ดังตารางที่ 4 และ 5 สำหรับ features นี้จะใช้ SyllableTokenizer จาก library nltk เพื่อช่วยในการแยก n-gram แต่ว่า SyllableTokenizer ไม่ได้รองรับภาษาไทย 100% ทำให้มีการแบ่งผิดพลาดอยู่บ้าง เช่น Muay ของ 2-gram ดังตารางที่ 5

Firstname	1-gram
Yuwadee	{'1grams-1': 'yu', '1grams-2': 'wa', '1grams-3': 'dee'}
Muay	{'1grams-1': 'mua', '1grams-2': 'y'}
Passorn	{'1grams-1': 'pas', '1grams-2': 'sorn'}
Nut	{'1grams-1': 'nut'}
Jan	{'1grams-1': 'jan'}
Janista	{'1grams-1': 'ja', '1grams-2': 'nis', '1grams-3': 'ta'}
Rungtip	{'1grams-1': 'rung', '1grams-2': 'tip'}
Bank	{'1grams-1': 'bank'}
Satit	{'1grams-1': 'sa', '1grams-2': 'tit'}
Mayla	{'1grams-1': 'may', '1grams-2': 'la'}

ตารางที่ 4 ตัวอย่างตารางกลุ่มของ 1-gram

Firstname	2-gram
Yuwadee	{'2grams-1': "('yu', 'wa')", '2grams-2': "('wa', 'dee')"}
Muay	{'2grams-1': "('mua', 'y')"}
Passorn	{'2grams-1': "('pas', 'sorn')"}
Nut	{}
Jan	{}
Janista	{'2grams-1': "('ja', 'nis')", '2grams-2': "('nis', 'ta')"}
Rungtip	{'2grams-1': "('rung', 'tip')"}
Bank	{}
Satit	{'2grams-1': "('sa', 'tit')"}
Mayla	{'2grams-1': "('may', 'la')"}

ตารางที่ 5 ตัวอย่างตารางกลุ่มของ 2-gram

3.5 DictVectorizer

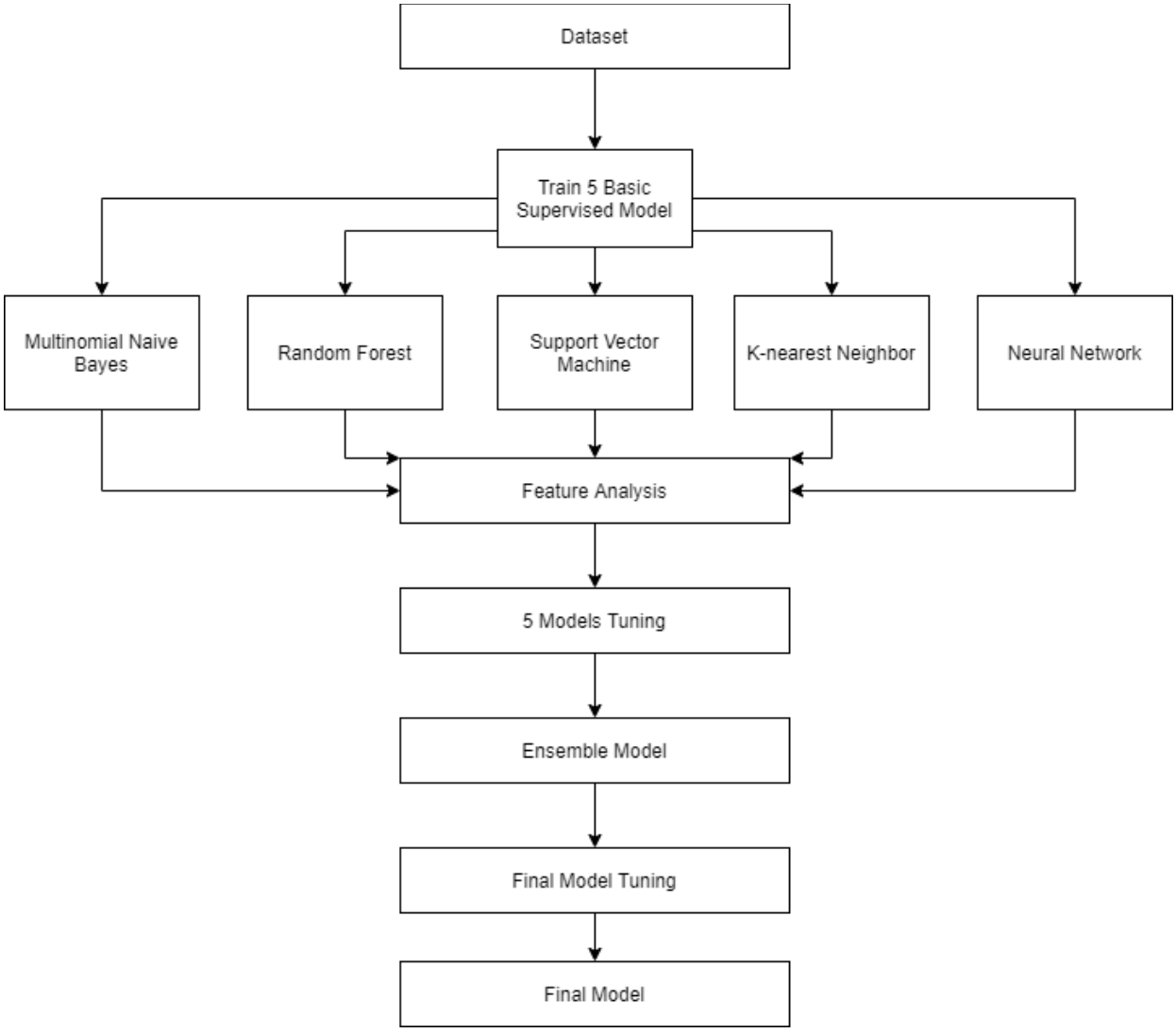
สำหรับโมเดลใดๆแล้ว เราไม่สามารถนำค่าประเภทอื่นๆนอกจาก float หรือ int เพื่อนำไปสร้าง model ได้ ดังนั้นจึงจำเป็นต้องแปลง feature ให้อยู่ในรูปที่ใช้งานได้ก่อน แต่เนื่องจาก feature มีจำนวนเยอะมาก การสร้าง column ใหม่จะเป็นการใช้พื้นที่เยอะ การใช้ dictvectorizer จะเป็น feature ที่อยู่ในรูป dict ให้เป็น sparse matrix ซึ่งจะลดพื้นที่จัดเก็บได้มากและสามารถนำไปเข้าโมเดลได้

3.6 Evaluation Approach

ในการวัดผลความแม่นยำ เราจะใช้ k-fold stratified cross validation ซึ่งเป็นวิธีที่จะแบ่งข้อมูลเท่าๆกันออกเป็น k ชุดโดยในแต่ละชุดจะมี training set จำนวน k-1 ชุดและ training set 1 ชุด แล้วนำมาทำนายผลแล้วเก็บไว้ ทำซ้ำไปเรื่อยๆจนครบ k ชุดแล้วนำค่าที่เก็บไว้มาหาค่าเฉลี่ยซึ่งข้อดีของวิธีนี้คือจะช่วยลด selection bias และ overfitting ได้ ยิ่งค่า k มากขึ้นยิ่งช่วยลดได้มากขึ้น แต่ก็จะใช้เวลาในการประมวลผลนานขึ้นด้วย

ค่า accuracy ในตารางทุกค่าที่เราบันทึกจะเป็นค่าที่เกิดจาก 10-fold stratified cross validation เสมอยกเว้นตอนใช้ RandomizedSearchCV ที่จะใช้เป็น 3-fold stratified cross validation เนื่องจากถ้าใช้ 10-fold จะใช้เวลาในการค้นหานานเกินไป

3.7 ขั้นตอนการสร้างโมเดล



รูปที่ 1 ขั้นตอนในการสร้างโมเดล

3.7.1 Train 5 Basic Supervised Model

ในขั้นตอนนี้เป็นการสร้างโมเดลแบบพื้นฐานโดยไม่ได้กำหนด hyperparameter เฉพาะเจาะจงโดยโมเดลที่สร้างขึ้นมาจะเป็น supervised model ได้แก่ Multinomial Naïve Bayes, Random Forest, Support Vector Machine, K-nearest Neighbor, Neural Network

3.7.2 Feature Analysis

เป้าหมายของขั้นตอนนี้คือหา features ที่ดีที่สุดสำหรับโมเดล 5 ตัว ในขั้นตอน Feature Analysis เราได้พิจารณาถึง features ดังต่อไปนี้ คือ

1.Substring 2.Character frequency 3.Syllable 1-gram 4.Syllable 2-gram

โดยมีข้อกำหนดว่า Syllable 1-gram กับ Syllable 2-gram จะไม่นำมาเป็น feature ร่วมกัน

เพื่อพิจารณาว่า feature ใดจะช่วยเพิ่มประสิทธิภาพของโมเดลได้ดีที่สุดเราจึงได้ทำการทดลองแบ่งกลุ่มเป็น 11 กลุ่มโดยแต่ละกลุ่มจะใช้โมเดลทั้ง 5 ตัว ได้แก่ Multinomial Naïve Bayes, Random Forest, Support Vector Machine, K-nearest Neighbor, Neural Network

หลังจากนั้นจะนำ accuracy มาหาค่าเฉลี่ยจากโมเดลทั้ง 5 ตัว โดยพิจารณาจาก feature ใดที่ให้ค่าเฉลี่ยสูงสุดเราจะเลือกใช้ feature นั้นเพราะโมเดลสุดท้ายที่เราจะสร้างนั้นเกิดจาก accuracy ของทั้ง 5 โมเดล ซึ่งผลลัพธ์ที่ออกมาเป็นดังตารางที่ 6

Features	Multinomial Naïve Bayes	Random Forest	Support Vector Machine	K-Nearest Neighbor	Neural Network	**ค่าเฉลี่ย**
Substring เท่านั้น	0.7808	0.7828	0.7830	0.7675	0.7748	0.7778
Character Frequency เท่านั้น	0.6243	0.7348	0.6309	0.7085	0.6929	0.6783
1-gram เท่านั้น	0.7577	0.7484	0.7609	0.7137	0.7516	0.7465
2-grams เท่านั้น	0.6846	0.6861	0.6964	0.5475	0.6945	0.6618
Substring+ Character Frequency	0.7815	0.7908	0.7853	0.7667	0.7790	0.7835
Substring+1-gram	0.7871	0.7861	0.7857	0.7760	0.7825	0.7825
Substring+2-gram	0.7876	0.7848	0.7879	0.7670	0.7853	0.7825
Character Frequency + 1-grams	0.7593	0.7721	0.7734	0.7440	0.7624	0.7622
Character Frequency+2-grams	0.7177	0.7439	0.7314	0.7154	0.7279	0.7273
Substring +Character Frequency+1-gram	0.7877	0.7890	0.7874	0.7711	0.7802	0.7831
Substring +Character Frequency+2-gram	0.7892	0.7890	0.7902	0.7656	0.7895	0.7847

ตารางที่ 6 ตารางแสดงค่า Accuracy ในช่วง features analysis

จากผลลัพธ์พบว่า Substring + Character Frequency+2-grams มีค่าเฉลี่ยสูงสุดคือ 0.7847 เราจึงเลือกใช้ feature นี้เพื่อไปทำในขั้นตอนต่อไปคือ Model Tuning

3.7.3 5 Model Tuning

ในขั้นตอน Model Tuning จะใช้ RandomizedSearchCV ซึ่งเป็น library ที่ช่วยในการทดสอบโมเดลเพื่อหา accuracy จาก parameter ที่เราใส่เข้าไป RandomizedSearchCV จะสุ่ม parameter เพื่อนำไปสร้างโมเดลที่มี hyperparameter ตามที่สุ่มไว้แล้วทดสอบหา accuracy โดยทั้งนี้กำหนดให้มี n_iter คือ 100 และ cv = 3 เพื่อไม่ให้ใช้เวลาในการสุ่มนานมากเกินไป

สำหรับ parameter ที่เราใช้ใน RandomizedSearchCV สำหรับ model แต่ละตัวเป็นดังต่อไปนี้

Classifier	Multinomial Naïve Bayes
Parameter ที่ใช้ใน RandomizedSearchCV	{'alpha': [1e-4,1e-3,1e-2,1e-1,1]}
Parameter ที่ได้จาก RandomizedSearchCV	{'alpha': 0.1}
Accuracy ที่ได้หลังจาก Model Tuning	0.791
Accuracy ก่อน Model Tuning	0.789
Accuracy ที่เพิ่มขึ้น	0.003

ตารางที่ 7 แสดง Model Tuning ของ Mutinomial Naïve Bayes

Classifier	Random Forest
Parameter ที่ใช้ใน RandomizedSearchCV	{'bootstrap': [True, False], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None], 'max_features': ['auto', 'sqrt'], 'min_samples_leaf': [1, 2, 4], 'min_samples_split': [2, 5, 10], 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
Parameter ที่ได้จาก RandomizedSearchCV	{'n_estimators': 2000, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 100, 'bootstrap': False}
Accuracy ที่ได้หลังจาก Model Tuning	0.795
Accuracy ก่อน Model Tuning	0.789
Accuracy ที่เพิ่มขึ้น	0.006

ตารางที่ 8 แสดง Model Tuning ของ Random Forest

Classifier	Support Vector Machine
Parameter ที่ใช้ใน RandomizedSearchCV	{'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001], 'kernel': ['linear','rbf', 'poly', 'sigmoid']}
Parameter ที่ได้จาก RandomizedSearchCV	{'kernel': 'rbf', 'gamma': 0.01, 'C': 10}
Accuracy ที่ได้หลังจาก Model Tuning	0.799
Accuracy ก่อน Model Tuning	0.790
Accuracy ที่เพิ่มขึ้น	0.009

ตารางที่ 9 แสดง Model Tuning ของ Support Vector Machine

Classifier	K-Nearest Neighbor
Parameter ที่ใช้ใน RandomizedSearchCV	{'n_neighbors' : list(range(1,31)), 'weights' : ['uniform','distance'], 'metric': ['minkowski','euclidean','manhattan']}
Parameter ที่ได้จาก RandomizedSearchCV	{'weights': 'distance', 'n_neighbors': 24, 'metric': 'manhattan'}
Accuracy ที่ได้หลังจาก Model Tuning	0.775
Accuracy ก่อน Model Tuning	0.765
Accuracy ที่เพิ่มขึ้น	0.010

ตารางที่ 10 แสดง Model Tuning ของ K-nearest Neighbor

Classifier	Neural Network
Parameter ที่ใช้ใน RandomizedSearchCV	{'hidden_layer_sizes': [(10,), (10,2), (10,4,2)], 'activation': ['identity', 'logistic', 'tanh', 'relu'], 'solver': ['sgd', 'adam'], 'alpha': [1e-4, 1e-3,1e-2,1e-1,1], 'learning_rate': ['constant', 'invscaling', 'adaptive']}
Parameter ที่ได้จาก RandomizedSearchCV	{'solver': 'adam', 'learning_rate': 'constant', 'hidden_layer_sizes': (10,), 'alpha': 0.1, 'activation': 'logistic'}
Accuracy ที่ได้หลังจาก Model Tuning	0.794
Accuracy ก่อน Model Tuning	0.789
Accuracy ที่เพิ่มขึ้น	0.005

ตารางที่ 11 แสดง Model Tuning ของ Neural Network

3.7.4 Ensemble Model

ในขั้นตอนนี้เป็นขั้นตอนเพื่อนำโมเดลทั้ง 5 ตัวมารวมกันเพื่อให้ทำนายได้แม่นขึ้น โดยใช้ Output ของทั้ง 5 โมเดลเป็น Input สำหรับ Neural Network ซึ่งจะให้ output ออกมาเป็นเพศที่เราทำนาย โดยผลลัพธ์ของ accuracy ที่ได้เป็นดังตารางที่ 12

Classifier	Multinomial Naïve Bayes	Random Forest	Support Vector Machine	K-Nearest Neighbor	Neural Network	Ensemble Model
Accuracy	0.791	0.795	0.799	0.775	0.794	0.921

ตารางที่ 12 ตารางแสดงค่า Accuracy ของทุกโมเดลหลังจากปรับ Hyperparameter แล้ว

3.7.5 Final Model Tuning

ในขั้นตอนสุดท้ายเพื่อเพิ่มประสิทธิภาพของโมเดลให้ได้มากที่สุด จึงได้ทำ Model Tuning อีกครั้ง

Classifier	Neural Network
Parameter ที่ใช้ใน RandomizedSearchCV	{'hidden_layer_sizes': [(10,), (10,2), (10,4,2)], 'activation': ['identity', 'logistic', 'tanh', 'relu'], 'solver': ['sgd', 'adam'], 'alpha': [1e-4, 1e-3,1e-2,1e-1,1], 'learning_rate': ['constant', 'invscaling', 'adaptive']}
Parameter ที่ได้จาก RandomizedSearchCV	{'solver': 'adam', 'learning_rate': 'adaptive', 'hidden_layer_sizes': (), 'alpha': 0.1, 'activation': 'tanh'}
Accuracy ที่ได้หลังจาก Model Tuning	0.923
Accuracy ก่อน Model Tuning	0.921
Accuracy ที่เพิ่มขึ้น	0.002

ตารางที่ 13 แสดง Model Tuning ของ Final Model

5.ข้อสรุปและข้อเสนอแนะ

โมเดลสุดท้ายที่สร้างขึ้นมามีความแม่นยำมากถึง 92.3% ซึ่งสามารถสรุปขั้นตอนการทำโมเดลได้ดังนี้ ขั้นที่หนึ่งคือ feature analysis ขั้นตอนนี้เป็นหนึ่งในขั้นตอนที่สำคัญที่สุดเพราะว่าจะส่งผลกระทบต่อขั้นตอนถัดๆไปด้วย ซึ่งถ้าไม่เลือกให้ดี โมเดลที่สร้างมาภายหลังก็จะไม่ดีตามไปด้วย ขั้นที่สองคือ 5 Model Tuning เพื่อเพิ่มประสิทธิภาพของ 5 โมเดลที่แตกต่างกัน ขั้นที่สามคือ Ensemble Model เป็นขั้นตอนการรวมโมเดลเข้าไว้ด้วยกันด้วยการใช้ Neural Network ขั้นตอนสุดท้ายคือ Final Model Tuning เพื่อเพิ่มประสิทธิภาพให้ได้มากที่สุด

จากประสิทธิภาพของโมเดลนี้จะช่วยให้นำไปใช้ในการออกนโยบายทางการตลาดกับกลุ่มลูกค้าที่ปกปิดเพศของตนได้ซึ่งจะมีช่วยเพิ่มยอดขายได้

เนื่องจากฐานข้อมูลที่ได้ทำการเก็บมากนั้นยังมีขนาดเล็กอยู่นั้นซึ่งมีจำนวน 10,031 หมายความว่าถ้าหากมีฐานข้อมูลที่ใหญ่มากขึ้นก็จะช่วยให้โมเดลมีประสิทธิภาพมากขึ้นด้วย และ Syllable n-gram บางชื่อยังมีการแบ่งที่ผิดพลาด เพราะว่า library nltk ไม่ได้รองรับการออกเสียงไทย 100% ถ้าสามารถแบ่งพยางค์ได้ตามการออกเสียงภาษาไทยได้แล้วผลลัพธ์น่าจะดีขึ้น