MBTI-PREDICTION

FINAL PROJECT

members: 蕭皓澤 顏愷威 謝承憲 許禎勻

0. MBTI

- MBTI (Myers-Briggs Type Indicator) : personality assessment tool that categorizes individuals into 16 personality types

0. MBTI

- based on preferences in four dichotomies :

  - Extraversion (E) vs. Introversion (I)

  - Sensing (S) vs. Intuition (N)

  - Thinking (T) vs. Feeling (F)

  - Judging (J) vs. Perceiving (P)

0. GOAL

- The goal of this project is to develop an accurate and effective predictive model for identifying **MBTI personality types** based on textual data.

CONTENT

01

TRAIN DATA

Data Source

Training Details

Data Preparation

Model Training

Preliminary Result

02

SEPARATE & REDUCE

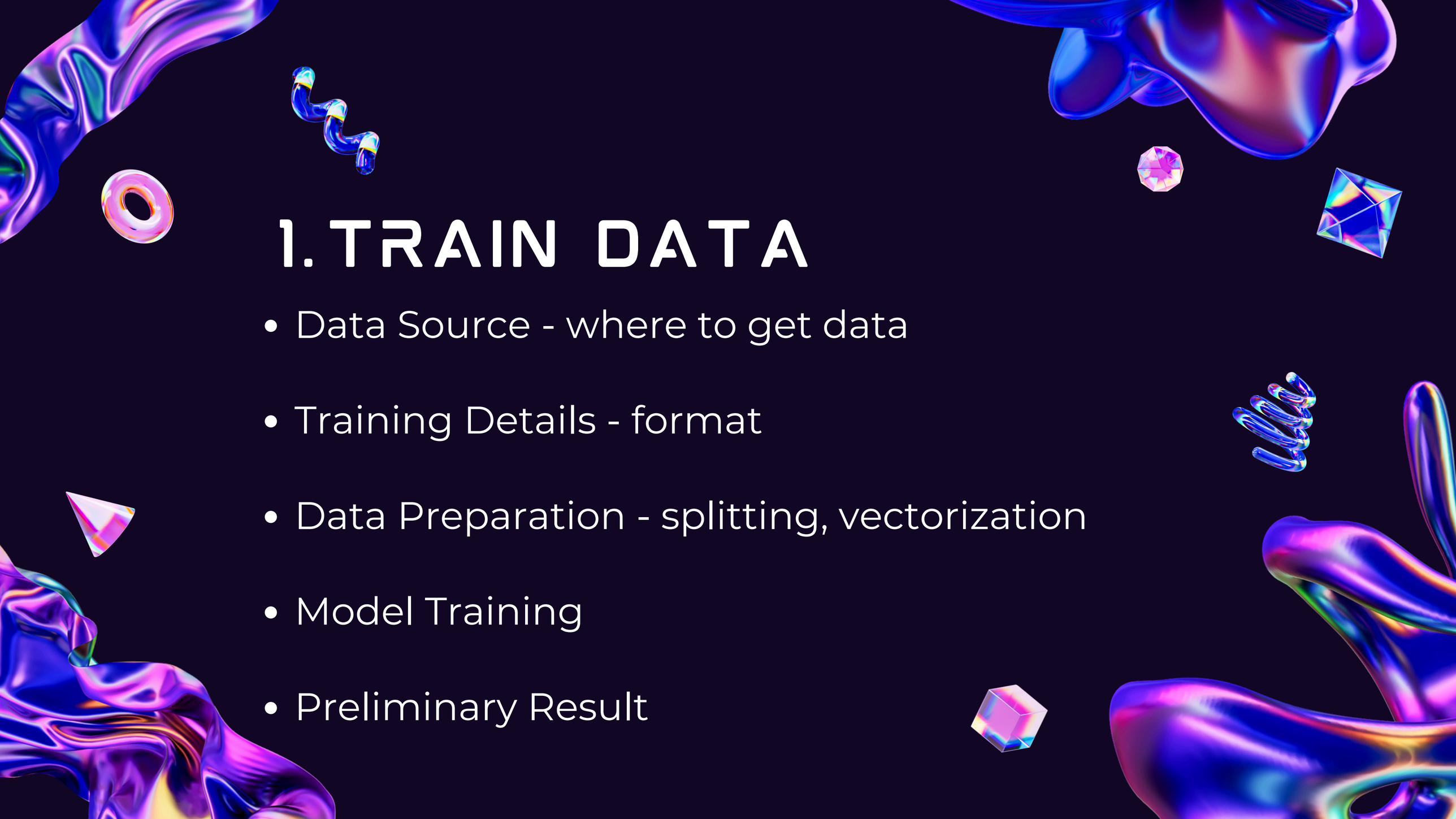
Two strategies to improve model performance

03

MERGE MODELS

Extract the best-performing models with the best accuracy

Weighed confidence score



1. TRAIN DATA

- Data Source - where to get data
- Training Details - format
- Data Preparation - splitting, vectorization
- Model Training
- Preliminary Result

WHERE TO GET DATA

Link : https://www.kaggle.com/datasets/zeyadkhalid/mbti-personality-types-500-dataset/data

KAGGLE

Kaggle is a data science competition platform and online community for data scientists and machine learning practitioners under Google LLC.

CONTENT

~106K preprocessed posts and their authors' personality types.
Posts are equal-sized: 500 words per sample.

DETAILS OF TRAINING

DATA FORMAT

content = text + “,” (comma) + MBTI type

```
feel definitely able maintain friendship must let know able live,INTJ
```

DATA SPLITTING

train set : test set = 9 : 1

```
# 將資料分為訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42, stratify=y)
```

TEXT VECTORIZATION

transforms textual data into numerical representations

MODEL TRAINING

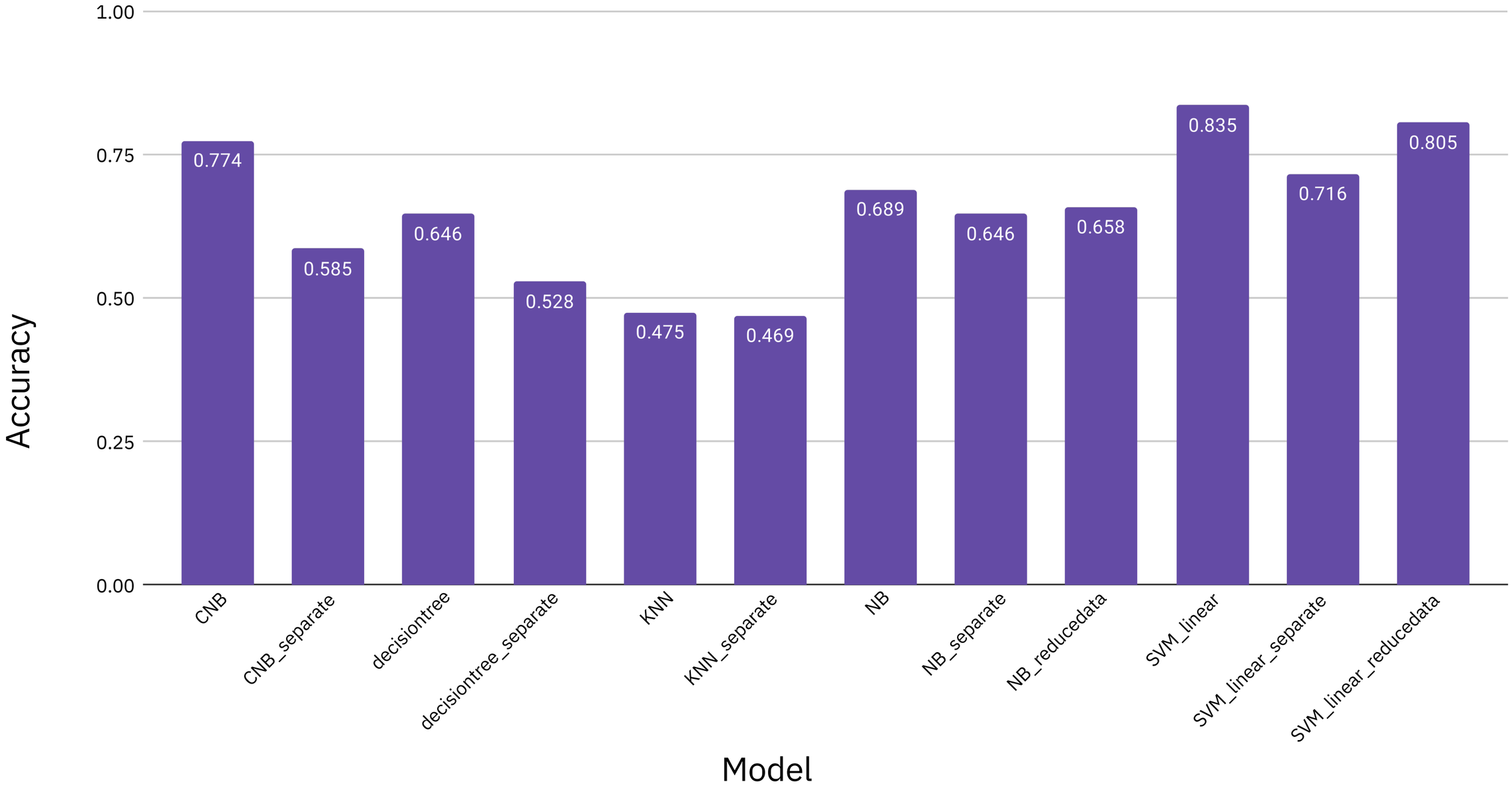
using different model training data, and measure our models by precision value, recall value, f1-score.

```
# 使用 CountVectorizer 處理 Naive Bayes 的資料
cnb_vectorizer = CountVectorizer()
X_train_cnb = cnb_vectorizer.fit_transform(X_train)
X_test_cnb = cnb_vectorizer.transform(X_test)
```

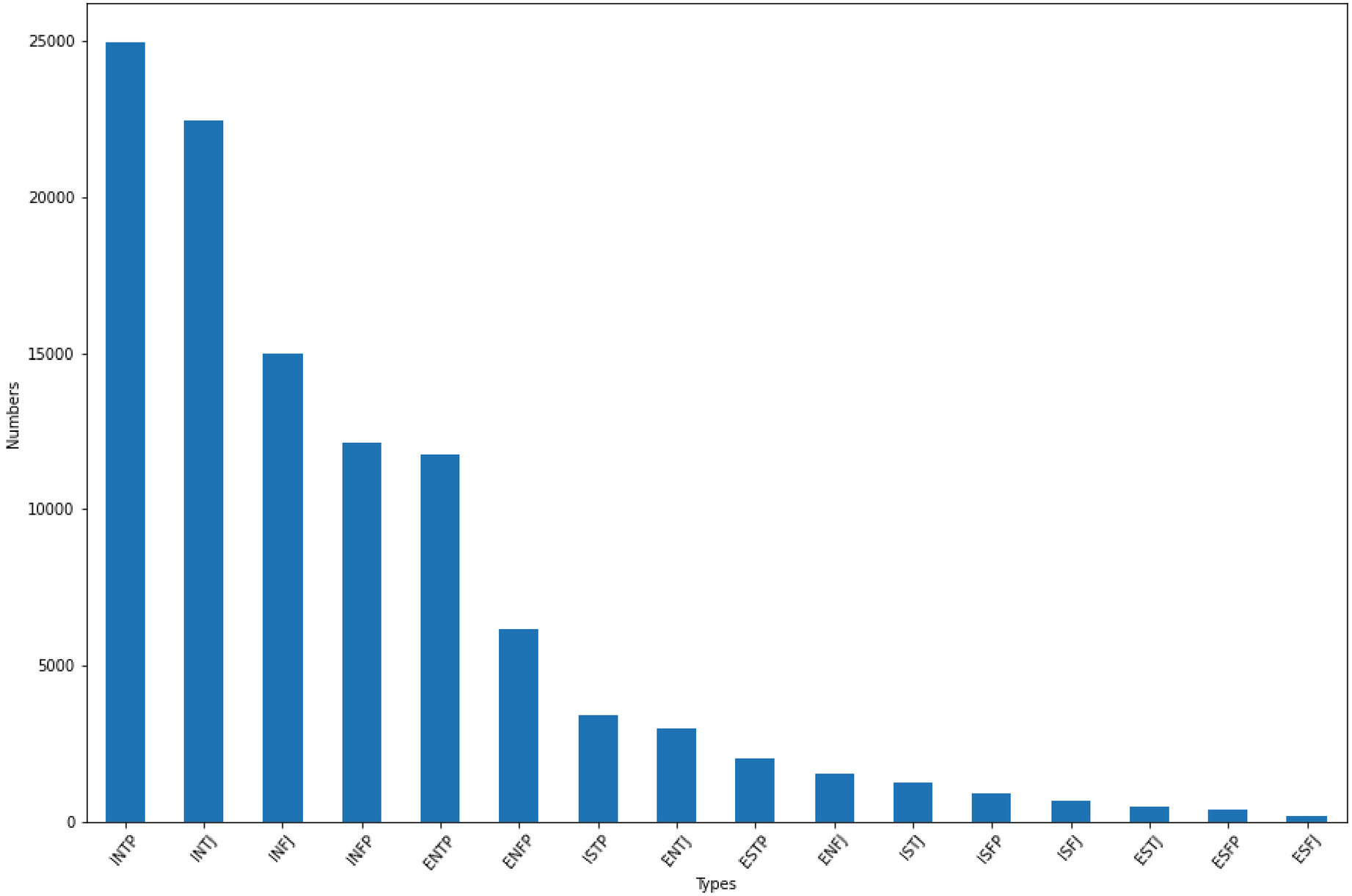
```
# 訓練 Complement Naive Bayes 模型
cnb_model = ComplementNB()
cnb_model.fit(X_train_cnb, y_train)
```

PRELIMINARY RESULTS

MBTI prediction accuracy of models



of different types of MBTI in Dataset



2.SEPARATE & REDUCE

- Two strategies to improve model performance

SEPARATE

IDEA

- MBTI has 16 categories, making it a complex multi-class classification problem.
- Breaking down multi-class into Binary Classifications?

APPROACH

- Divide the MBTI prediction task into 4 independent binary classification problems (I/E, S/N, T/F, J/P)
- Train separate models for each dimension to predict its binary outcome.
- Combine predictions from all 4 dimensions to reconstruct the full MBTI type.

CODE SNIPPET

```python
# 將資料分為訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42, stratify=y)

# 定義 MBTI 四個字母的位置
mbti_letters = ["I/E", "N/S", "T/F", "J/P"]

# 儲存四個獨立的 SVM 模型
models = []

# 訓練四個模型
for i in range(4):
    # 抽取第 i 個字母作為目標
    y_train_binary = [label[i] for label in y_train]
    model = LinearSVC(random_state=42, class_weight="balanced", C=0.75)
    model.fit(X_train, y_train_binary)
    models.append(model)

# 測試階段
predictions = []
for i, model in enumerate(models):
    # 針對測試資料預測第 i 個字母
    letter_predictions = model.predict(X_test)
    predictions.append(letter_predictions)

# 組合四個模型的結果為完整的 MBTI 類型
final_predictions = ["".join(letters) for letters in zip(*predictions)]
```

REDUCE

IDEA

- MBTI classification suffers from an imbalanced dataset
- Adjust the sample size of every MBTI type?

APPROACH

- Aligning the sample sizes of all classes

CODE SNIPPET

```
# 下採樣訓練資料
rus = RandomUnderSampler(random_state=42)
X_train_resampled, y_train_resampled = rus.fit_resample(X_train, y_train)
```

EXAMPLE-SVM LINEAR

- SVM linear 0.8418
- SVM linear separate 0.71.93
- SVM linear reduce 0.8052

CONCLUSION

- Both separate and reduce method perform worse than the original method
- Possible reasons:
  - The four MBTI dimensions are not completely independent.(separate)
  - Each binary classifier introduces its own errors. The compounded errors can lead to lower overall accuracy.
  - Some MBTI types have too few samples.(reduce)

3.MERGE MODELS

- Extract the best-performing models with the best accuracy
- Weighed confidence score

THE TWO BEST MODELS

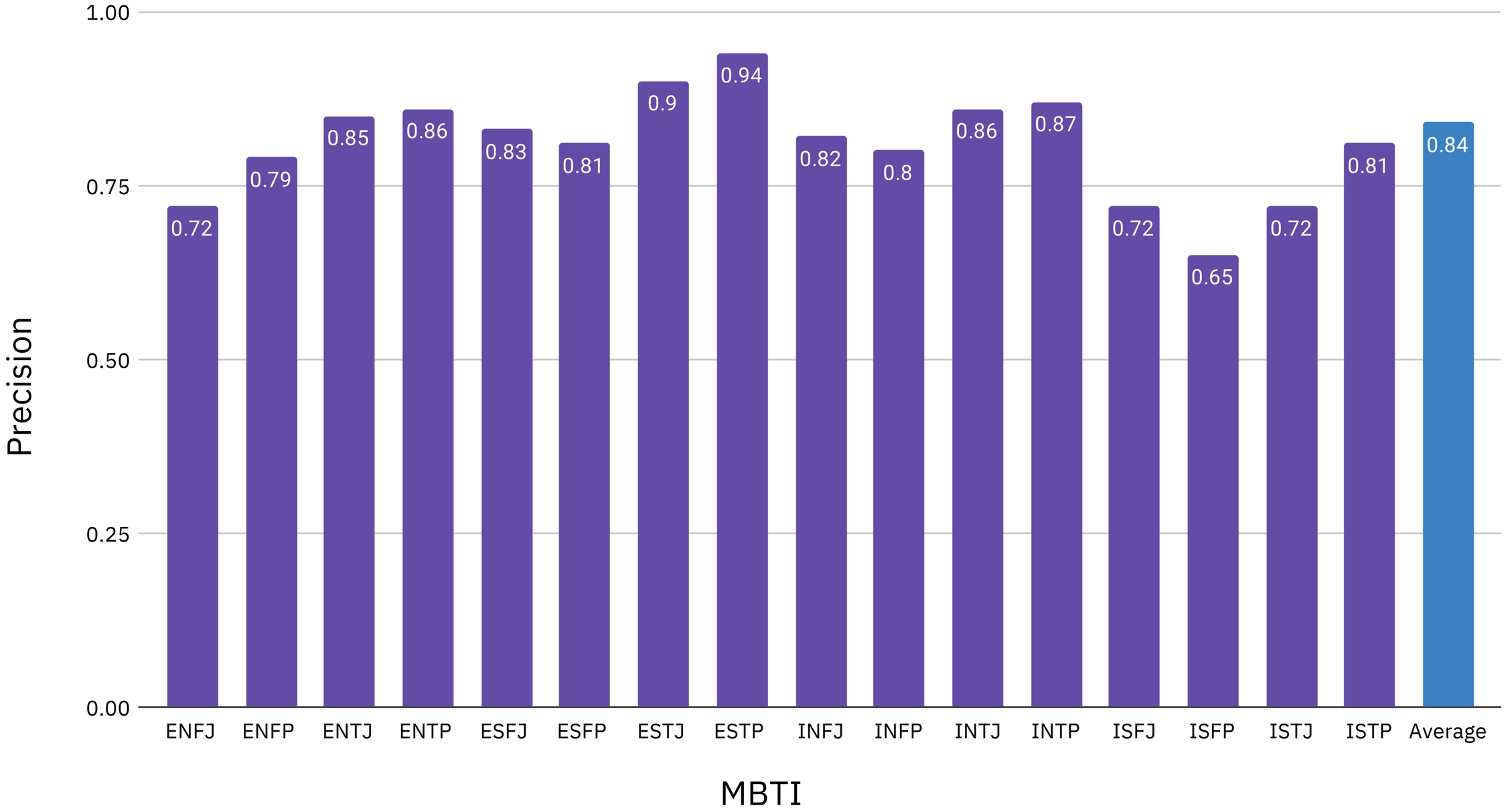
- SVM (Support Vector Machine) -Linear
  - Accuracy : 0.835
- CNB (Complement Naive Bayes)
  - Accuracy : 0.774

HOW TO MERGE?

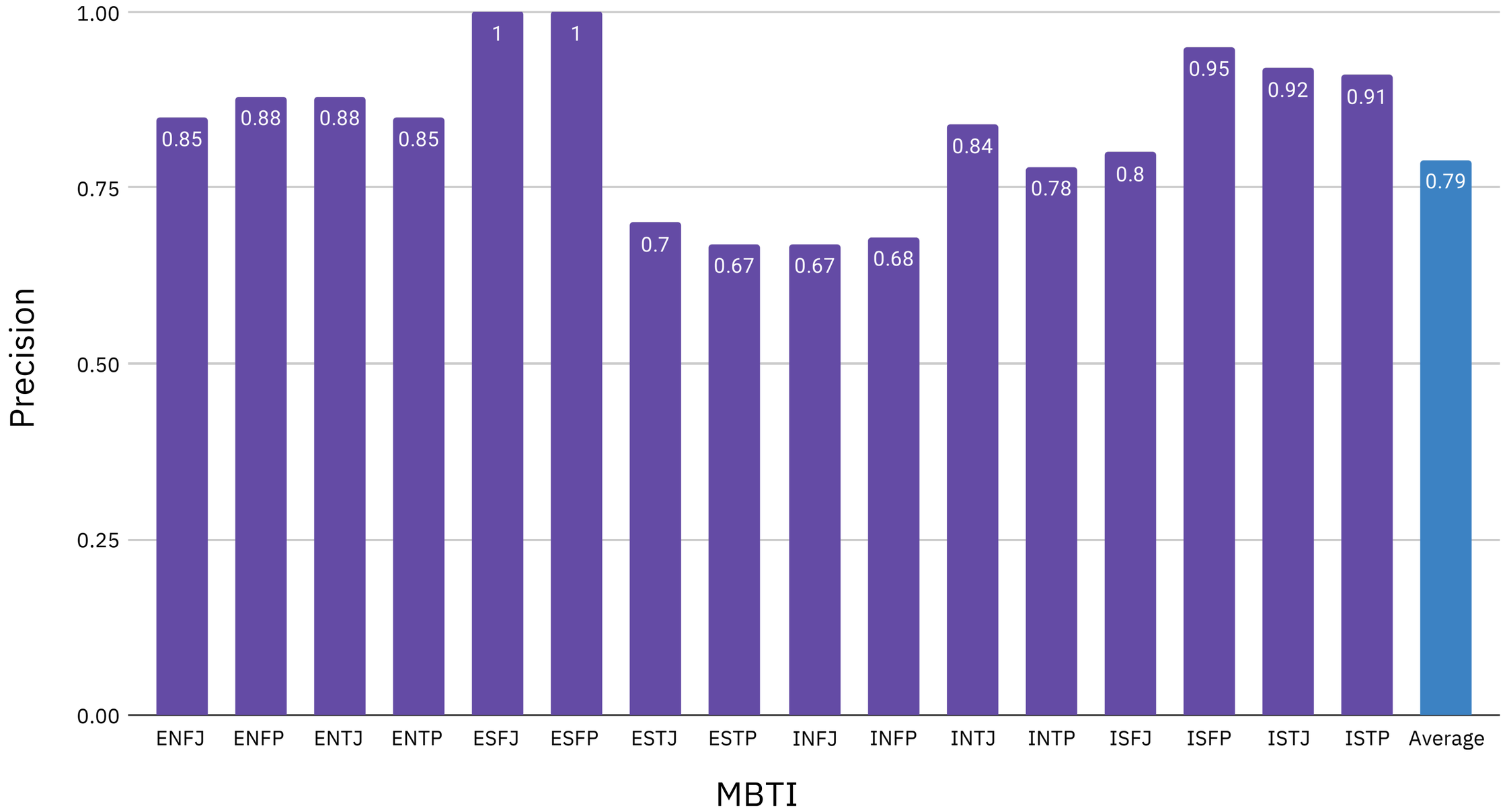
$$Precision = \frac{TP}{TP + FP}$$

- The two models produce the confidence scores of each MBTI type.
- We weight the confidence scores of each MBTI type by multiplying them with the corresponding **precision** values of the two models from previous results.
- Confidence scores from CNB and SVM models are combined using a weighted average, with equal weightage (0.5) assigned to both models.
- Finally, to predict the final MBTI type, we return the one that has the largest value of the confidence score.

Precision of SVM_linear



Precision of CNB



CODE SNIPPET

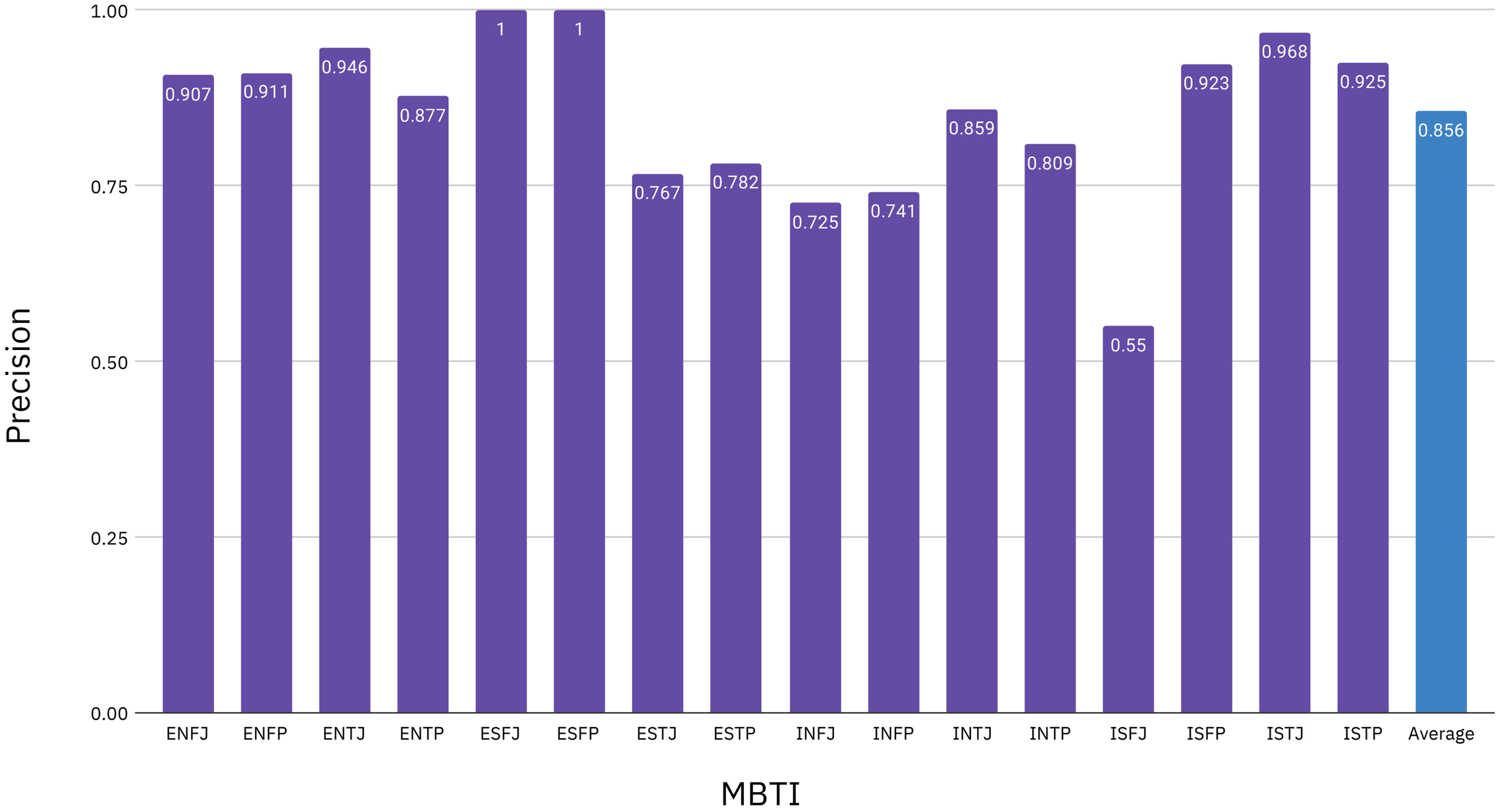
Based on the precision value

```
# 定義每個 label 的權重
label_weights_cnb = { "ENFJ": 0.85 ,"ENFP": 0.88 , "ENTJ":0.88 , "ENTP":0.84 , "ESFJ":1.00 , "ESFP":1.00, "ESTJ":0.63 , "ESTP":0.62,"INFJ":0.67,"INF
label_weights_svm = { "ENFJ": 0.75 ,"ENFP": 0.81 , "ENTJ":0.82 , "ENTP":0.84 , "ESFJ":0.69 , "ESFP":0.77, "ESTJ":0.83 , "ESTP":0.92,"INFJ":0.84,"INF
```

```
# 對每個樣本，將 CNB 和 SVM 的信心分數加權處理
nb_score = {label: score * label_weights_cnb[label] for label, score in zip(cnb_labels, cnb_scores[i])}
svm_score = {label: score * label_weights_svm[label] for label, score in zip(cnb_labels, svm_scores[i])}

# 合併 NB 和 SVM 的分數
total_score = {label: 0.5 * nb_score[label] + 0.5 * svm_score[label] for label in cnb_labels}
```

Precision of merged model (CNB and SVM_linear)



THANKS

Work Distribution	
蕭皓澤	Python code & Model training
顏愷威	Data organization, chart creation
謝承憲	Data organization, presentation creation
許禎匀	Presentation creation

MBTI Predict

APPENDIX（CHINESE VERSION）

1. 輸出機率：在先前的結果可以看出 SVM(linear kernel) 以及CNB 已經有不錯的準確度，而兩個模型都可以對每個類別提供「信心分數」
 - 但是兩者的準確度在不同類型上相差甚遠(ex.ISTJ 由 SVM 預測的話只有 0.65 , 但CNB 有 0.95)

2. 尋找權重：我們採用的方法是將每一類別的信心分數 * 先前測試的準確率，期望能使 SVM , CNB 都能夠保留（提供較大成分）較能夠準確預估的類別

final score of class c = SVM 對 class c 的信心分數 * SVM 對 class c 預估的準確率
+ CNB 對 class c 的信心分數 * CNB 對 class c 預估的準確率

APPENDIX（CHINESE VERSION）

3. 輸出最有信心的類別：輸出經過權重加總後信心分數最大者
  - 盡量保留兩個模型優勢的類別
  ex.ISTJ 以 SVM 預測的話只有 0.65 , 但CNB 有 0.95，那麼最後成績就是 SVM score * 0.65 + CNB * 0.95，CNB 應會主導分數

4. 最終結果：以macro-average 來評估，可發現整合後的模型為0.83，超過了原有的0.81(SVM)以及 0.78(CNB)