Personality Analysis Through Social Media Based on Machine Learning

1st Rofiq Fadli Nurrohman
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
rofiqfadli@student.telkomuniversity.ac.id

2nd Kris Sujatmoko
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
krissujatmoko@telkomuniversity.ac.id

3nd Thomhert Suprapto
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
thomhert@telkomuniversity.ac.id

Abstrak - Seringkali, dalam dunia profesional, sulit untuk memastikan bahwa keterampilan, kepribadian, dan pekerjaan sesuai satu sama lain. Ketidaksesuaian ini dapat menyebabkan kinerja buruk dan menyulitkan karyawan untuk memaksimalkan potensi mereka. Salah satu faktor yang menyebabkan ketidaksesuaian ini adalah jumlah pekerjaan yang tidak sesuai dengan kepribadian seseorang. Untuk mengatasi masalah ini, meningkatkan kesadaran diri adalah kunci untuk menjadi lebih analitis terhadap potensi dan sifat Anda. Meningkatkan kesadaran diri akan membantu Anda menemukan kekuatan dan kelemahan pribadi Anda. Program yang dapat menilai kemampuan seseorang dan memberikan saran karir yang sesuai dapat membantu pengenalan diri ini. Alat penemuan diri yang tersedia di situs web PSYCHEE membantu Anda mengidentifikasi kepercayaan diri dan potensi Anda. Data pengguna yang berkualitas kemudian dapat digunakan oleh program untuk membuat rekomendasi pekerjaan. Hasil pengujian menunjukkan bahwa model XGBoost memiliki akurasi paling tinggi. Situs web PSYCHEE menawarkan akurasi prediksi sekitar 89% untuk klasifikasi teks X atau Twitter, dan akurasi 100% untuk kumpulan data yang diproses menggunakan perintah obrolan GPT. Model yang dikembangkan digunakan untuk mengkategorikan kepribadian MBTI berdasarkan konten media sosial. Untuk melakukan fungsinya, aplikasi web Psyche digunakan.

Kata kunci: Ketidakcocokan pekerjaan, kepribadian, PSYCHEE, XGBoost, aplikasi Psyche

Abstract In the workplace, it can be challenging to make sure that a person's work, personality, and skills complement one another. Employees may perform poorly as a result of this mismatch and find it challenging to reach their full potential. The amount of jobs that don't fit a person's personality is one thing that contributes to this mismatch. The solution to this issue is to become more analytical about your abilities and characteristics by developing your self-awareness. Gaining more self-awareness will assist you in identifying your own advantages and disadvantages. Programs that evaluate skills and offer appropriate career guidance might support this process of self-discovery. The PSYCHEE website offers a self-discovery tool that assists you in determining your potential and level of confidence. The computer can then utilize qualified user data to recommend jobs. The XGBoost model offers the highest accuracy, according to the test findings. For text classification on Twitter or X, the prediction accuracy provided by the PSYCHEE website is approximately 89%, whereas for data sets processed with GPT chat commands, the accuracy is 100%. MBTI personalities are categorized using the created model according to social media material. The Psyche online application was utilized in order for it to function.

Keywords: Job mismatch, personality, PSYCHEE, XGBoost, Psyche app

I. INTRODUCTION

The mismatch or gap between work possibilities and industrial demands, which is brought about by a mismatch in education and skills, is one of the problems in Indonesia's employment sector [1]. Regardless of their college major, a large number of Indonesian graduates of higher education were compelled to look for work outside of their field of study due to an imbalance between supply and demand for employment. This situation, known as a "horizontal mismatch," has grown to be a serious problem, especially for recent college grads, as there are more graduates than there are spots available in some majors. Many graduates are forced to choose jobs unrelated to their majors as a result [2]. An employee must work harder to acquire the skills or competences required for the job when they operate in an area that does not align with their educational background, which is why a mismatch between education and employment is problematic. Employees may feel uneasy at work and have low job satisfaction as a result of having to learn new things and delve into material that is different from what they already know. Reduced productivity and business expansion will also result from this [3]. NLP is defined as "patterns or programming created from the relationship between the brain (neuro), language (linguistic), and body state" in the Encyclopedia of Systemic NLP and NLP New Coding [4]. Researchers can better understand public opinion by employing NLP approaches to investigate people's nuanced perspectives and attitudes about the designated health services [5]. Lack of excellent human resources is one of the reasons for the mismatch, which is partly brought on by people's ignorance of themselves. One tactic for building human resources is self-development [6].

Understanding one's own character, often referred to as self-awareness, may help one grow their self-awareness and become more aware of their personality, limitations, and potential. Because of this, someone may utilize self-evaluation as an endeavor or step toward improving their own

quality and ultimately turning into a quality human resource [7]. Paul Costa and Robert McCrae developed the Big Five personality theory, which is one of the most well-known approaches in personality psychology. The five fundamental aspects of human nature that are highlighted by this theory are neuroticism, agreeableness, conscientiousness, extraversion, and openness to new experiences. Individual personality differences may be defined by a variety of qualities that are represented by each of these categories. The extraordinary validity of the Big Five hypothesis in many cultural contexts and situations has been shown via extensive empirical testing, making it an important tool in personality studies [8].

II. THEORY REVIEW

A. Myers-Briggs Type Indicator (MBTI)

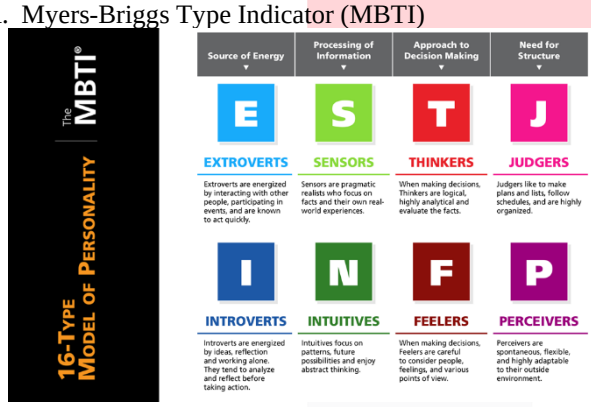

Figure 1. MBTI Personality

Based on Carl Jung's theory of psychological types, Isabel Briggs Myers and her mother Katharine Cook Briggs created the popular Myers-Briggs Type Indicator (MBTI), a tool for assessing personality. Based on four dichotomies—Extraversion (E) vs. Introversion (I), Sensing (S) vs. Intuition (N), Thinking (T) vs. Feeling (F), and Judging (J) vs. Perceiving (P)—the MBTI assigns people to one of 16 personality types. It is commonly used to assist people in understanding their preferences and communication styles in a variety of contexts, such as organizational behavior, counseling, and personal development. The MBTI has been criticized for its validity and reliability in predicting psychological characteristics and behavior, despite its widespread use [9].

B. GPT-3.5-Turbo-0125

A variant of OpenAI's Generative Pre-trained Transformer (GPT) model is called GPT-3.5-Turbo-0125. It makes use of a Transformer architecture designed to understand and produce text in challenging circumstances. GPT-3.5-Turbo-0125 can generate extremely natural and helpful text in a variety of situations because it was trained on a sizable dataset. Compared to earlier models, this one requires less training time because to OpenAI's optimization and computationally effective techniques. The main features of this model are its ability to produce text that is very realistic and diversified, as well as its excellent scalability, which enables it to be used in applications with real-time requirements. The model is very beneficial in practical applications like virtual assistants and content generation because of its remarkable ability to generalize across many text sorts and conditions, despite its enormous complexity that requires significant processing resources [10].

C. XGBoost

Extreme Gradient Boosting, or XGBoost, is a boosting algorithm renowned for its quickness and potency. It functions by gradually constructing an ensemble of decision trees, with each new tree fixing the mistakes of the preceding tree. Regularization techniques are used by XGBoost to prevent overfitting and enhance overall model performance. XGBoost is usually regarded as efficient, while the training time may vary based on the amount of the dataset and the model's complexity. Because of its excellent scalability, the model can operate steadily on big datasets. In addition to having outstanding generalization capabilities, XGBoost is well-known for being simple to use in a wide range of machine learning applications because of its wealth of documentation and vibrant community. To get the best results, thorough hyperparameter tuning is required, and more iterations and larger data volumes may result in longer training times [11].

D. CatBoost

Yandex created CatBoost (Categorical Boosting), a boosting method that can handle categorical data without requiring a lot of setup, like one-hot encoding. CatBoost handles category data using certain techniques that lead to more accurate and efficient processing. Because of the computational approaches used and the optimization of the algorithm, CatBoost frequently requires less training time than other boosting models. Because it is simpler to build and apply than some other boosting models, this model is less complex. CatBoost can handle large datasets with consistent performance and has high generalization capabilities. CatBoost is a great option for many different jobs because of its great use in machine learning applications, especially when dealing with categorical data [12].

E. Gradient Boosting

Gradient boosting is a boosting technique that gradually constructs a prediction model. To minimize the error of past models, every new model is trained using a gradient-based optimization technique. Gradient boosting is consequently highly useful for problems related to regression and classification. One of gradient boosting's most important characteristics is its capacity to handle a wide range of input sources and produce extraordinarily accurate models. However, the length of the training period may vary based on the size of the dataset and the number of iterations. The hyperparameters need to be carefully tuned in order to minimize overfitting and obtain optimal performance. Although gradient boosting can be difficult to scale and can only handle huge datasets with unstable performance, this

model's complexity makes it a powerful option for a wide range of machine learning applications [13].

III. RESEARCH METHODS

The 1-5 scale point provides a formal manner to compare and assess models based on a variety of essential criteria, including generalization ability, complexity, scalability, ease of use, and training time. Using this scale, we can more objectively assess the strengths and shortcomings of each model, allowing us to make more educated decisions about which model is best suited to the application's demands.

Table 1 Solution Analysis and Selection

Selection Criteria	Point	GPT 3.5		XGBoost	
		Rating	Point Value	Rating	Point Value
Data Training Time	20%	3	0,6	5	1
Complexity	20%	2	0,4	3	0,6
Generalization Ability	30%	5	1,5	4	1,2
Scalability	20%	3	0,6	5	1
Usability	10%	4	0,4	4	0,4
Total Score		3,5		4,2	
Rating		4		1	
Continue?		**NO**		**YES**	

Gradien Boost		CatBoost	
Rating	Point Value	Rating	Point Value
3	0,6	5	1
3	0,6	2	0,4
4	1,2	4	1,2
5	1	5	1
4	0,4	4	0,4
3,8		4	
3		2	
NO		**NO**	

- **Data Training Time**
  - GPT-3.5-Turbo-0125: 3 - The training time of this model is relatively short thanks to the optimizations performed, but it requires large computational resources for the training process [10].
  - XGBoost: 5 - Highly efficient in training time, enabling fast model training even with large datasets [11].
  - Gradient Boosting: 3 - Training time can be long depending on the size of the dataset and the number of iterations required to achieve convergence [12].
  - CatBoost: 5 - Fast in training, thanks to optimization techniques and good efficiency in processing categorical data [13].
- **Complexities**
  - GPT-3.5 Turbo-0125: 5: This model is extremely complicated, utilizing the Transformer design, which needs extensive expertise and computational resources. This paradigm is used for large-scale data management and processing [10].
  - XGBoost: 3 - Although XGBoost is effective, its complexity is moderate. Users must adjust hyperparameters to maximize model performance, but the documentation and supportive community make it easier to manage [11].
  - Gradient Boosting: 4 - Requires a good understanding of the training process and hyperparameter adjustments. This complexity can provide difficulties in implementation and tweaking [12].
  - CatBoost: 3 - It is less complex than GPT-3.5-Turbo-0125 and Gradient Boosting, owing to the built-in support for category data, making it easier to use [13].
- **Generalization Ability**
  - GPT-3.5-Turbo-0125: 5 - The model has a very high generalization ability thanks to training with a wide and diverse dataset. This allows the model to handle various topics and contexts well, producing relevant and consistent text in a variety of situations [10].
  - XGBoost: 5 - XGBoost demonstrates strong generalization capabilities by effectively handling large and complex data. The boosting technique used minimizes errors and improves model performance on unseen data [11].
  - Gradient Boosting: 5 - Gradient Boosting offers high generalization capabilities because of its incremental model building and iterative error reduction. This enables it to adapt to various sorts of data and make accurate predictions [12].
  - CatBoost: 5 - CatBoost provides a high level of generalization, particularly for categorical data. A specific technique used to analyze categorical data without extensive preprocessing allows the model to adapt to a variety of data situations [13].
- **Scalability**
  - GPT-3.5-Turbo-0125: 4 - Provides acceptable scalability for large-scale applications but necessitates extensive computer infrastructure. This methodology is intended to manage vast amounts of data and produce consistent outcomes [10].
  - XGBoost: 5 - Highly scalable; can handle big datasets with high performance. The methods utilized provide efficiency and efficacy in large data settings [11].
  - Gradient Boosting: 4 - Scalability is good, however training time may be longer on larger datasets. The methodology remains successful on huge datasets, however scalability is influenced by data quantity and complexity [12].
  - CatBoost: 5 - Highly scalable due to algorithm optimization and support for categorical data. This enables the model to run efficiently on huge datasets [13].
- **Usability**
  - GPT-3.5-Turbo-0125: 4 - OpenAI's API makes it relatively simple to use, but maximizing its benefits in actual applications needs a thorough grasp of how it works [10].
  - XGBoost: 4 - Easy to use with good documentation and community. However, hyperparameter calibration may require time and additional effort [11].

- Gradient Boosting: 3: The requirement to change several hyperparameters and comprehend the complicated training process might have an impact on ease of use [12].
- CatBoost: 4 - Simple to use, especially with built-in support for classified data, which eliminates the need for substantial data preparation [13].

A. Machine Learning Model

The first implementation step is to select the machine learning algorithm that will be used for the model. The model with the highest accuracy will be shown on the website. Model building options include the following machine learning algorithms:

1. Gradient Boosting
2. XGBoost
3. CatBoost
4. GPT, However, API keys are necessary for fine-tuning, and token limitations influence the cost of training and running the model.

Before training and testing the model, a dataset including text and an MBTI personality type label is required. The dataset used in this test is a composite dataset obtained from Kaggle.

The obtained dataset is subsequently moved to the preparation step. During this stage, the raw datasets will be cleaned, tokenized, lemmatized, vectorized, and separated into training and test data.

For classification testing, the machine learning models Gradient Boosting, XGBoost, and CatBoost were applied. The model was trained using an alternative dataset for this test. There are 100,867 data points in the dataset for this exam, comprising text and 16 MBTI personality types. There were 100,867 data points in total, with 90% of the data being used for training and 10% for testing. With this split ratio, the model can utilize the largest possible portion of the training data while maintaining sufficient evaluation capabilities to gauge the model's generalization. This model was trained using a preprocessed dataset. The model's hyperparameters are in the default range for this test.

The model chosen based on accuracy and training duration will be adjusted by adjusting the hyperparameters to get the highest accuracy. The hyperparameters for this test will just be iteration /n_estimator, max depth, and learning rate. The three parameters were chosen because they have a major influence on the model's performance.

IV. RESULT AND ANALYSIS

The code in the picture below imports all of the libraries needed to clean the data and launches the "WordNetLemmatizer" object, which lemmatizes and saves a list of stopwords collected from the English language.

Figure 1 libraries required to clean the data

```
import re
import pandas as pd
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import nltk
from tqdm import tqdm

# Download stopwords
nltk.download('stopwords')

# Initialize the lemmatizer and stopwords list once to optimize performance
lemmatizer = WordNetLemmatizer()
useless_words = stopwords.words("english")
```

The code shown in the figure below removes superfluous components from the raw dataset, tokenizes each word into tokens separated by spaces, and then lemmatizes each word.

Figure 2 cleans the raw dataset

```
unique_type_list = [x.lower() for x in ['INFJ', 'ENTP', 'INTP', 'INTJ', 'ENTJ', 'ENFJ', 'INFP', 'ENFP',
                                        'ISFP', 'ISTP', 'ISFJ', 'ISTJ', 'ESTP', 'ESFP', 'ESTJ', 'ESFJ']]

def pre_process_sentence(sentence, remove_stop_words=True, remove_mbti_profiles=True):
    # Remove URL links
    sentence = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', ' ', sentence)
    # Remove non-words and excess spaces
    sentence = re.sub(r'[^a-zA-Z]+', ' ', sentence).lower()
    # Remove repeating letters
    sentence = re.sub(r'([a-z])\1{2,}', '', sentence)
    # Tokenize, remove stop words, and lemmatize
    words = sentence.split()
    if remove_stop_words:
        words = [word for word in words if word not in useless_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    sentence = ' '.join(words)
    # Remove MBTI personality words if required
    if remove_mbti_profiles:
        sentence = re.sub(r'\b(?:' + '|'.join(unique_type_list) + r')\b', '', sentence)

    return sentence

def clear_text(data, remove_stop_words=True, remove_mbti_profiles=True):
    cleaned_text = []
    for sentence in tqdm(data['posts']):
        # Preprocess the sentence using the pre_process_sentence function
        cleaned_sentence = pre_process_sentence(sentence, remove_stop_words, remove_mbti_profiles)
        cleaned_text.append(cleaned_sentence)
    return cleaned_text
```

The code in the picture below divides the data in a 90:10 ratio and utilizes the scikit-learn library's "TfidfVectorizer()" for text and "LabelEncoder()" for labels to convert the cleaned dataset to numbers. Because the dataset is sufficiently large (100,867 data points), a data split ratio of 90% for training and 10% for testing was adopted. The model will train on 90% of the data, ensuring that there is enough to gather and learn patterns from. Using 10% of the total data as test data guarantees that the model has enough information to evaluate its performance on data that is not readily apparent.

Figure 3 Cleaned dataset

```
# Baca data
df = pd.read_csv('myml_100k.csv')

x = df['posts']
y = df['cleanlab_corrected_label']

# Menggunakan TfidfVectorizer untuk mengubah teks menjadi fitur
vectorizer = TfidfVectorizer()
X_vect = vectorizer.fit_transform(x)

# Membagi data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X_vect, y, test_size=0.1, random_state=42)

# Mengubah label menjadi bentuk numerik
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

After preprocessing, the data is ready for training the machine learning model. The models to be trained are first imported and started before being trained using the training data. In the first test, all models were trained using training data with the default hyperparameter settings.

Figure 4 Gradient Boosting Model

```
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier()
model.fit(X_train, y_train)
```

```
GradientBoostingClassifier()
```

The code in the image below starts the gradient boosting classifier model and trains it using the training data. After training, "classification_report" and "accuracy_score" are used to display the classification report of the test data and calculate the model's accuracy. The model accuracy was 87.81%, with a total training time of 15.5 hours.

```
from sklearn.metrics import classification_report, accuracy_score
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
accuracy = accuracy_score(y_test, y_pred)
accuracy_percentage = accuracy * 100
print(f'Accuracy: {accuracy_percentage:.2f}%')
```

```
              precision    recall  f1-score   support

        ENFJ       0.00      0.00      0.00        12
        ENFP       0.83      0.82      0.82       555
        ENTJ       0.83      0.79      0.81       236
        ENTP       0.86      0.84      0.85      1058
        ESFJ       0.00      0.00      0.00         5
        ESFP       0.00      0.00      0.00         3
        ESTJ       0.80      0.40      0.53        10
        ESTP       0.76      0.57      0.65       129
        INFJ       0.88      0.89      0.89      1522
        INFP       0.89      0.87      0.88      1247
        INTJ       0.89      0.90      0.90      2368
        INTP       0.89      0.92      0.91      2561
        ISFJ       0.00      0.00      0.00         7
        ISFP       0.00      0.00      0.00         5
        ISTJ       0.75      0.74      0.74        65
        ISTP       0.86      0.81      0.84       304

    accuracy                           0.88     10087
   macro avg       0.58      0.54      0.55     10087
weighted avg       0.88      0.88      0.88     10087

Accuracy: 87.81%
```

Figure 5 XGBoost model

```
# Melatih model XGBoost
model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', n_jobs=-1)
model.fit(X_train, y_train_encoded)
```

```
d:\Project 8\NewPrompt\myenv\Lib\site-packages\xgboost\core.py:158: UserWarning: [13:35:31] WARNING:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=None,
              n_jobs=-1, num_parallel_tree=None, objective='multi:softprob', ...)
```

The code in the image below fulfills the same purpose as the previous code: to train the model and display the classification result report from the XGBoost model. The model accuracy achieved was 89.06%, with a total training duration of 54 minutes.

Figure 6 The XGBoost accuracy and training duration

```
y_pred = model.predict(X_test)

print(classification_report(y_test_encoded, y_pred))
accuracy = accuracy_score(y_test_encoded, y_pred)
accuracy_percentage = accuracy * 100
print(f'Accuracy: {accuracy_percentage:.2f}%')
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.86      0.83      0.84       555
           2       0.84      0.80      0.82       236
           3       0.87      0.86      0.87      1058
           4       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         3
           6       0.50      0.20      0.29        10
           7       0.78      0.68      0.73       129
           8       0.90      0.91      0.90      1522
           9       0.89      0.89      0.89      1247
          10       0.91      0.91      0.91      2368
          11       0.89      0.93      0.91      2561
          12       0.00      0.00      0.00         7
          13       0.00      0.00      0.00         5
          14       0.74      0.71      0.72        65
          15       0.89      0.83      0.86       304

    accuracy                           0.89     10087
   macro avg       0.57      0.53      0.55     10087
weighted avg       0.89      0.89      0.89     10087

Accuracy: 89.06%
```

Figure 7 CatBoost model's

```
# Melatih model CatBoost
model = CatBoostClassifier(eval_metric='MultiClass', thread_count=-1, verbose=15, iterations=100)
model.fit(X_train, y_train_encoded)
```

```
Learning rate set to 0.5
0:      learn: 1.2761419        total: 43.7s    remaining: 1h 12m 10s
15:     learn: 0.4878535        total: 11m 57s  remaining: 1h 2m 45s
30:     learn: 0.3937274        total: 22m 48s  remaining: 50m 45s
45:     learn: 0.3869656        total: 33m 13s  remaining: 39m
60:     learn: 0.3842167        total: 43m 42s  remaining: 27m 56s
75:     learn: 0.3772150        total: 54m 9s   remaining: 17m 6s
90:     learn: 0.3759637        total: 1h 4m 34s        remaining: 6m 23s
99:     learn: 0.3733183        total: 1h 10m 49s       remaining: 0us

<catboost.core.CatBoostClassifier at 0x27d6dcbc4a0>
```

The code below does the same function as the previous code: it trains the model and presents the CatBoost model's classification result report. The model accuracy was 85.73%, with a total training time of 1.12 hours.

Figure 8 The CatBoost accuracy and training duration

```
y_pred = model.predict(X_test)

print(classification_report(y_test_encoded, y_pred))
accuracy = accuracy_score(y_test_encoded, y_pred)
accuracy_percentage = accuracy * 100
print(f'Accuracy: {accuracy_percentage:.2f}%')
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
           1       0.80      0.78      0.79       555
           2       0.80      0.71      0.76       236
           3       0.83      0.84      0.83      1058
           4       0.00      0.00      0.00         5
           5       0.00      0.00      0.00         3
           6       0.60      0.30      0.40        10
           7       0.68      0.57      0.62       129
           8       0.86      0.88      0.87      1522
           9       0.86      0.86      0.86      1247
          10       0.87      0.88      0.88      2368
          11       0.88      0.91      0.89      2561
          12       0.00      0.00      0.00         7
          13       0.00      0.00      0.00         5
          14       0.76      0.52      0.62        65
          15       0.82      0.76      0.79       304

    accuracy                           0.86     10087
   macro avg       0.55      0.50      0.52     10087
weighted avg       0.85      0.86      0.85     10087

Accuracy: 85.73%
```

The XGBoost model outperforms the other two models in terms of training time and accuracy. As a consequence, the XGBoost model was chosen for classification and implementation on the website. The previously trained XGBoost model can be tuned again by adjusting its hyperparameters. To get the best accuracy, the model will be retrained with different hyperparameter values. In this test, the hyperparameter values "learning_rate" and "max_depth" will be assessed to determine the difference in model accuracy results. In this test, the "n_estimators" model goes through 100 iterations.

n_estimators	learning_rate	max_depth	training time	accuracy
100	0.1	6	47m5s	88.69 %
100	0.2	6	48m7s	88.97 %
100	0.3	6	38m22 s	89.06 %
100	0.4	6	37m4s	89.08 %
100	0.5	6	38m19 s	88.63 %
100	0.4	7	47m7s	88.77 %
100	0.4	5	28m55 s	89.13 %
100	0.4	4	23m15 s	89.55 %
100	0.4	3	18m22 s	89.35 %

After testing, the model with the best accuracy achieved 89.55%. The model that has been trained and obtained the maximum accuracy is saved in a file with the pickle(.pkl) extension. The model will then be used to perform MBTI personality classification activities on the website.

V. CONCLUSION

The trained model's performance and outcomes will also be affected by the hyperparameters that are set. Using a dataset of 100,867 observations, the model was first trained using the default hyperparameters for this test. Based on this test, the XGBoost model, after 54 minutes of training, gets the greatest accuracy of 89.06%. The XGBoost model was selected for use after its accuracy and training time were evaluated. The XGBoost model is first fine-tuned by retraining and hyperparameter modification. The model that tested the best had an accuracy of 89.55%.

REFERENCE

[1] L. Mahdiah, “Ketidaksesuaian antara Pendidikan dan Kebutuhan Tenaga Kerja Masih Besar,” databoks. Diakses: 22Oktober 2023. [Daring]. Tersediapada: https://databoks.katadata.co.id/datapublish/2019/05/28/ketidaksesuaian-antarapendidikan-dan-kebutuhan-tenaga-kerja-masih-besar

[2] A. P. Yonanda dan H. Usman, “Determinan Status Horizontal Mismatch pada Pekerja Lulusan Pendidikan Tinggi di Indonesia,” *Jurnal Ketenagakerjaan*, vol. 18, no. 2, hlm. 142–157, Agu 2023, doi: 10.47198/jnaker.v18i2.239.

[3] T. Hoturu, A. Dilly, G. Papuling, P. Studi, A. Bisnis, dan U. H. Namotemo, “Dampak Mismatch Pendidikan-Pekerjaan Terhadap Pengembangan Keahlian Karyawan Di Halmahera,” *Jurnal Ilmu Manajemen dan Akutansi*, vol. 10, no. 2, hlm. 102–103, 2022.

[4] R. B. Dilts and J. A. Delozier, The Encyclopedia of Systemic NLP and NLP New Coding. 2000

[5] D. L. Girsang, N. A. Sidiq, and N. T. S. Elenaputri, “Analisis Sentimen Masyarakat terhadap Layanan BPJS Kesehatan dan Faktor-Faktor Pendukung Opini dengan Pemodelan Natural Language Processing (NLP),” Emerging Statistics and Data Science Journal, vol. 1, no. 2, pp. 238–249, Jun. 2023, doi: 10.20885/esds.vol1.iss.2.art24.

[6] A. Saoqillah, “PENGEMBANGAN SDM MELALUI MANAJEMEN DIRI,” *Jurnal Riset Manajemen dan Bisnis (JRMB) Fakultas Ekonomi UNIAT*, vol. 1, no. 2, hlm. 145–152, Okt 2016, doi: 10.36226/jrmb.v1i2.18.

[7] I. Setyawati, “Self-Awareness Untuk Memperbaiki Diri dan Memperbaiki Hidup,” Binus University. Diakses: 23 Oktober 2023. [Daring]. Tersedia pada: https://binus.ac.id/binusian-journey/2022/08/08/self-awareness-untuk-memperbaikidiri-dan-memperbaiki-hidup/#:~:text=Kenapa sih self-awareness menjadi,sebagai langkah peningkatan kualitas diri

[8] T. Hoturu, A. Dilly, G. Papuling, P. Studi, A. Bisnis, dan U. H. Namotemo, “Dampak Mismatch Pendidikan-Pekerjaan Terhadap Pengembangan Keahlian Karyawan Di Halmahera Utara,” *Jurnal Ilmu Manajemen dan Akutansi*, vol. 10, no. 2, hlm. 102, 2022.

[9] I. Briggs-Myers and P. B. Myers, Gifts Differing: Understanding Personality Type. 1995. [Online]. Available: http://ci.nii.ac.jp/ncid/BA28259319.

[10] OpenAI, "GPT-3.5-Turbo Model Card," 2023. [Online]. Available: https://www.openai.com/research/gpt-3-5-turbo

[11] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016. [Online]. Available: https://arxiv.org/abs/1603.02754.

[12] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: unbiased boosting with categorical features," arXiv preprint arXiv:1810.11363, 2018. [Online]. Available: https://arxiv.org/abs/1810.11363.

[13] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, vol. 29, no. 5, pp. 1189-1232, 2001. [Online]. Available: https://projecteuclid.org/euclid.aos/1013203451.