# Generative AI Competition CS523 Summer-1 2023 Detect the AI-generated photos

**Hemant Kumar Singh**
BU ID: U17284611
Boston University
hks@bu.edu

## Abstract

As digital technologies advance, so does the ability to generate false identities. Sophisticated generative artificial intelligence (AI) can now fabricate convincing fake images, posing a serious threat to the integrity of online services and opening a path for fraudulent activities. This report describes the development of a robust machine learning algorithm devised to effectively counter this escalating digital deception. The algorithm has been trained on a broad dataset comprising both AI-generated and authentic images. Employing a model capable of distinguishing between real and artificially generated photos with remarkable precision, it provides a potent strategy to battle digital identity fraud. The performance of the algorithm illustrates significant improvements in online security, promising a safe environment for online users and secure transactions. This innovation offers a considerable potential for practical application; the envisioned integration into existing tech platforms can contribute substantially to the development of secure and reliable online services. By ensuring a safer digital world, the proposed solution affirms the transformative role of machine learning in combating the darker side of AI advancements.

## 1 Introduction

As artificial intelligence (AI) continues to evolve, its transformative power is evident across a myriad of sectors. However, it also presents notable challenges, a critical one being the creation of false identities, chiefly the generation of convincingly fake profile images. The sophistication of AI-based generative models, such as Generative Adversarial Networks (GANs), has reached a point where AI-generated human images are nearly indistinguishable from real ones, thereby escalating the potential for misuse.

Despite the enormity of the digital deception, the capacity of robust systems to effectively distinguish between real and artificially generated profile images is limited. Current traditional detection mechanisms and rule-based systems struggle against the escalating complexity of AI-generated fake images. Consequently, there is an urgent need for resilient, innovative solutions.

In response to this challenge, this research presents a robust machine learning algorithm specifically designed to distinguish between AI-generated and authentic images. This solution capitalizes on the strengths of various machine learning models to detect subtle patterns and discrepancies that can differentiate a real image from an AI-generated one.

This paper provides a comprehensive analysis of the proposed solution's design, implementation, and performance. The proposed solution underscores the transformative potential of machine learning to combat challenges accompanying AI advancements, thereby contributing to a safer digital world.

## 2 Methodology

### 2.1 Environment setup

- For this competition, Google colab has been used.
- Datasets: Train and test csv filesBigrit [2023] provided by the competition has been used.

#### 2.1.1 Libraries used

1. Numpy: Provides multi-dimensional array support for efficient computation, heavily used for matrix operations.
2. Pandas: Data manipulation and analysis tool. Provides data structures like DataFrame and Series for handling and visualizing data.
3. TensorFlow: Open-source library for creating and training Machine Learning and Deep Learning models.
4. Keras: High-level API for building, training, and evaluating neural network models, used as an interface for TensorFlow.
5. Scikit-learn (Sklearn): Comprehensive library for Machine Learning, provides functions for regression, classification, clustering, and preprocessing.
6. Imbalanced-learn (Imblearn): Offers re-sampling techniques for handling class imbalance, integrates well with Scikit-learn.
7. Matplotlib: Visualization library for creating static, animated, and interactive plots in Python.
8. XGBoost: Implements gradient boosting algorithm for regression and classification problems, renowned for performance and speed.

### 2.2 Exploratory data analysis

#### 2.2.1 Key findings

The two target classes are heavily imbalanced as shown in Fig. 1

- Class '0' i.e real data count: 3850.
- Class '1' i.e fake data count: 1400.

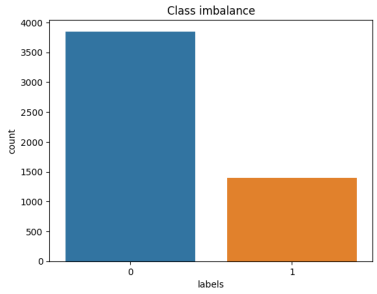


Figure 1: Target class imbalance distribution

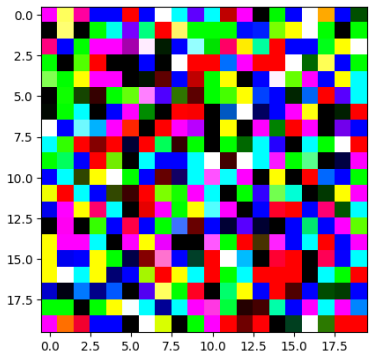


Figure 2: Original Image Reconstruction

Mean and Standard distribution check

- All features are approximately zero-centered (mean close to 0), as shown in Fig. 3.
- Standard deviations are close to 1, as shown in Fig. 4.

All the images in the datasets are 20 x 20 x 3 (channels). The underlying data is all processed and flattened to 1200 features, so they cannot be converted back to the original images. The result if we try to convert it back to the original image is shown in Fig. 2.

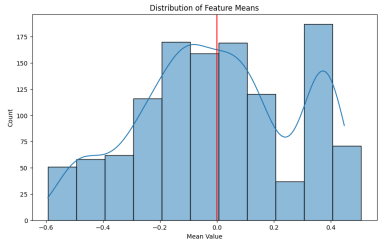


Figure 3: Mean distribution of features

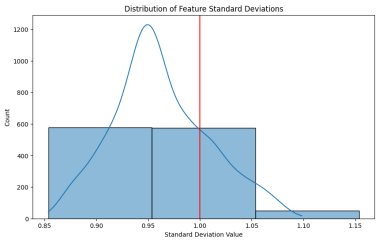


Figure 4: Standard deviation of features

### 2.2.2 Data preparation

1. SMOTE is used to balance the number of data of each class. There were 73% rows of real images and only 27% rows of fake images in the data set.
2. Verified that data is scaled correctly. The dataset was scaled as it was already processed data. All features are approximately zero-centered and standard deviations are close to 1.

### 2.2.3 Various ML Models and architectures considered for this task

1. Simple fully connected neural network (MLP)
2. Convolutional neural network
3. Transformer
4. Tabnet
5. K-Nearest neighbor
6. Random forest
7. XGBoost
8. SVM
9. Naive Bayes classifier
10. Ensemble of MLP and XGBoost
11. Ensemble of MLP, XGBoost, SVM
12. Ensemble of Naive Bayes classifier, KNN, XGBoost
13. Ensemble of KNN, XGBoost
14. Ensemble of MLP, Naive Bayes classifier, KNN, XGBoost
15. Ensemble of KNN, XGBoost, Gradient Boosting Classifier
16. Model stacking Naive Bayes classifier, KNN, XGBoost with meta learner as Logistic Regression
17. Model stacking Naive Bayes classifier, KNN, XGBoost with meta learner as Gradient Boosting Classifier
18. Model stacking KNN, XGBoost, Logistic Regression with meta learner as Naive Bayes classifier

### 2.2.4 Understanding the choice of specific ML models for this specific task

1. Multilayer Perceptron (MLP): Given the high-dimensionality of this data (1200 features per image), MLP, an artificial neural network, can capture the complex relationships between features. It's capable of learning non-linear decision boundaries which might be very useful as the image data can have complex pixel interactions that differentiate real from fake.
2. XGBoost: This model is an excellent choice due to its high performance, scalability, and ability to handle imbalanced datasets. XGBoost has built-in capabilities for performing weighted classification to help counteract the class imbalance in this data (more 'real' than 'fake' images).

3. K-Nearest Neighbors (KNN): The KNN algorithm could capture the local structure of your image data. This local structure could be unique to AI-generated images, thereby assisting in distinguishing them from real ones.
4. Support Vector Machines (SVM): SVMs can effectively handle high-dimensional data, which is perfect for this 1200-feature dataset. With the right kernel, SVMs can identify complex boundaries that separate AI-generated images from real ones.
5. Logistic Regression: This model provides a baseline performance metric. Given the binary nature of this problem (real vs. fake), Logistic Regression could help identify useful features and give insights about their relationship with the target variable.
6. Naive Bayes Classifier: This model is efficient with high-dimensional data and can provide a probabilistic framework for this task. While it assumes feature independence (which may not hold true for image data), it can still act as a good baseline model and sometimes provide surprisingly good results.
7. Gradient Boosting Classifier: This ensemble model will construct new trees that attempt to correct errors made by the previously trained trees. Given the complexity and potential non-linearities in this data, Gradient Boosting might be able to carve out the decision boundary effectively.
8. TabNet: A deep learning model designed for tabular data, TabNet uses an attention mechanism for dynamic feature selection, capturing complex interactions within your 1200-dimensional flattened image data. Its interpretability, offering feature importance scores, aids in understanding key distinguishing factors.

By creating an ensemble of these diverse models, we can leverage the strengths of each, potentially improving the ability to distinguish between AI-generated and real images. Each model might be able to pick up on different patterns or features within the images that could indicate whether they are real or fake. By combining these predictions, we could achieve higher overall accuracy.

#### 2.2.5 Progressive approach towards optimal model identification

1. Initial model exploration began with basic fully connected Multilayer Perceptron, yielding a modest Bitgrit score of 77%.
2. Attempted image restoration for CNN application was unsuccessful due to the processed nature of the images, deeming CNN ineffective for this task.
3. Experimentation with a transformer model resulted in a lower Bitgrit score of 0.64, indicating inadequate performance.
4. The classification-oriented KNN model demonstrated significant promise with a Bitgrit score of 95%.
5. Further exploration with other classification models, namely Random Forest, SVM, and XGBoost, did not surpass the KNN performance.
6. Literature review, focusing on a research paper by Shwartz-Ziv and Armon [2022], suggested superior results for tabular classification using a hybrid of deep learning and XGBoost.
7. Diversifying the model options by using ensemble methods with a variety of models: MLP, XGBoost, KNN, SVM, Logistic Regression, Naive Bayes Classifier, and Gradient Boosting Classifier.
8. Attempted data augmentation by reshaping data to the original image and applying transformations, but models were ineffective.
9. An ensemble model comprising KNN, XGBoost, and Naive Bayes Classifier obtained a high Bitgrit score of 95.8%.
10. Exploring model stacking architectures, the most effective combination was observed with base learners (KNN, Logistic Regression, XGBoost) and a Meta learner (Naive Bayes Classifier), reaching a Bitgrit score of 95.9%.
11. Tabnet, despite its known effectiveness, did not yield superior results in this context, leading to the final selection of the high-scoring model stack.

### 2.2.6 Description of best model architecture

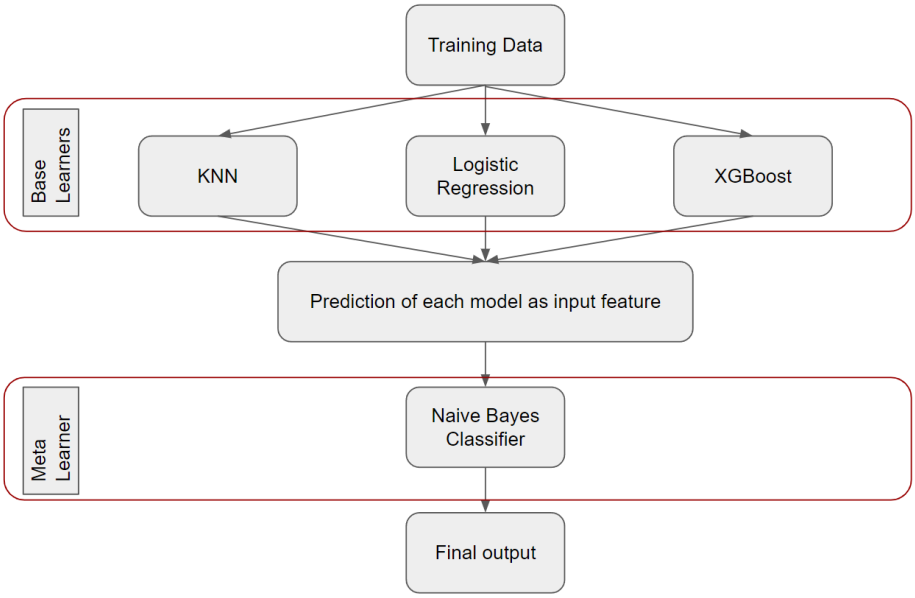


Figure 5: Model architecture diagram

As shown in Fig. 5, a Stacking Classifier is implemented to address the classification problem of identifying AI-generated photos from real ones. The Stacking Classifier is an ensemble learning technique that leverages the strengths of multiple models and combines their predictions using another model, known as the final estimator. This technique attempts to improve the predictive performance by exploiting the complementary predictive power of the different models.

For the initial models (base estimators), K-Nearest Neighbors (KNN), XGBoost (XGB), and Logistic Regression were used.

1. The KNN model was configured with the 'manhattan' metric and the hyperparameters include 15 neighbors, 'p' parameter set to 1, and 'uniform' weights. The 'manhattan' metric, also known as L1 distance, is useful in high dimensional spaces and can help the model navigate through the 1200-dimensional space effectively.
2. The XGB model, a gradient boosting algorithm, has its hyperparameters as subsample as 0.8, number of estimators as 100, min_child_weight as 3, max_depth as 5, learning_rate as 0.1, lambda as 1.5, gamma as 0.1, colsample_bytree as 0.8, and alpha as 0.1. It uses a binary logistic loss function. These hyperparameters are obtained through cross-validation and they ensure that the XGBoost model is able to capture complex patterns in the data without overfitting.
3. The Logistic Regression model uses default parameters and serves as a baseline model, providing linear decision boundaries.

These models are chosen due to their different decision-making strategies: KNN making decisions based on local structure, XGB being a powerful tree-based algorithm, and Logistic Regression providing a linear perspective. This diversity can enhance the robustness of the final model.

The final estimator of the stacking classifier is a Gaussian Naive Bayes model, which makes predictions based on the outputs of the initial models. This probabilistic model considers each input feature independently, reducing the potential of overfitting and adding another layer of diversity to the ensemble.

The stacking classifier is trained using 5-fold cross-validation, which helps ensure the robustness of the model and reduces the likelihood of overfitting.

Upon evaluation, the stacking classifier achieved an accuracy of 0.86952 and an F1 score of 0.74862 on the validation set, showing a good balance between precision and recall. The Bitgrit score of this competition is an impressive 0.95988. These scores demonstrate the effectiveness of the stacking classifier for this competition.

## 3 Result

Various models and their performances are shown in Fig. 6.

| Model | Architecture used | Validation set Accuracy | Validation set F1 score | Bitgrit score |
|---|---|---|---|---|
| 1 | Model stacking:<br>Base learners: KNN, Logistic Regression, XGBoost<br>Meta learner: Naive Bayes Classifier | 0.86952 | 0.74862 | 0.95988 |
| 2 | Ensemble:<br>KNN, XGBoost, Naive Bayes Classifier | 0.84285 | 0.73684 | 0.95833 |
| 3 | Model stacking:<br>Base learners: KNN,Naive Bayes Classifier, XGBoost<br>Meta learner: Logistic Regression | 0.9 | 0.78527 | 0.9553 |
| 4 | Model stacking:<br>Base learners: KNN, Naive Bayes Classifier, XGBoost<br>Meta learner: Gradient Boosting Classifier | 0.9 | 0.78439 | 0.95518 |
| 5 | KNN | 0.69545 | 0.76396 | 0.95238 |
| 6 | Ensemble:<br>MLP, Naive Bayes Classifier, KNN, XGBoost | 0.93051 | 0.93266 | 0.91083 |
| 7 | Ensemble:<br>KNN, XGBoost | 0.88761 | 0.7366 | 0.91228 |
| 8 | Ensemble:<br>KNN, XGBoost, Logistic Regression, Naive Bayes Classifier | 0.85809 | 0.85235 | 0.88693 |
| 9 | Ensemble:<br>KNN, XGBoost, Gradient Boosting Classifier | 0.88 | 0.71232 | 0.8812 |
| 10 | Ensebmle:<br>MLP and XGBoost | 0.9279 | 0.93 | 0.87719 |
| 11 | Model stacking:<br>Base learners: MLP and XGBoost<br>Meta learner: Logistic Regression | 0.9312 | 0.93 | 0.84977 |
| 12 | Ensemble:<br>MLP, XGBoost,SVM | 0.9429 | 0.94 | 0.84969 |
| 13 | Model stacking:<br>Base learners: KNN, XGBoost, Logistic Regression<br>Meta learner: Naive Bayes Classifier | 0.83142 | 0.66791 | 0.82727 |
| 14 | MLP | 0.84857 | 0.69126 | 0.77496 |
| 15 | Tabnet | 0.75047 | 0.76237 | 0.66993 |

Figure 6: Model performance on Bitgrit table

### 3.1 Evaluation metrics

The performance of the models was evaluated based on three metrics: validation set accuracy, validation set F1 score, and Bitgrit score.

1. Validation set accuracy measures how well the model performs on unseen data that was withheld during the training process. It quantifies the ratio of correct predictions to total predictions.
2. F1 score is the harmonic mean of precision and recall. It represents a balanced measure of the model's precision (correctly identified positive cases out of all identified positive cases) and recall (correctly identified positive cases out of all actual positive cases).
3. Bitgrit score is a unique scoring metric used for this specific competition. It measures the model's robustness and generalizability on the unseen data, taking into account the model's ability to handle variability and its performance stability.

### 3.2 Model Validation

In this context, the standout models were Model 1 and Model 6, each excelling in different metrics.

1. **Model 1**: The stacked model 1, combining KNN, Logistic Regression, XGBoost, and Naive Bayes, stands out with a Bitgrit score of 0.95988. The stacking method and the diversity of base learners likely contributed to the model's robustness and superior generalization on unseen data, making it an excellent choice for predictions on new data.

   As shown in Fig. 7, the confusion matrix indicates that the model correctly classified 709 real images and 204 AI-generated images, while it misclassified 61 real images as AI-generated and 76 AI-generated images as real.

   As shown in Fig. 8, the ROC AUC (Receiver Operating Characteristic Area Under Curve) for Model 1 is 0.935630797773655, indicating a strong ability to discriminate between classes, i.e., real and AI-generated images.

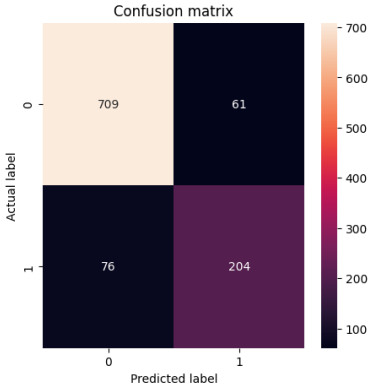


Figure 7: Confusion matrix

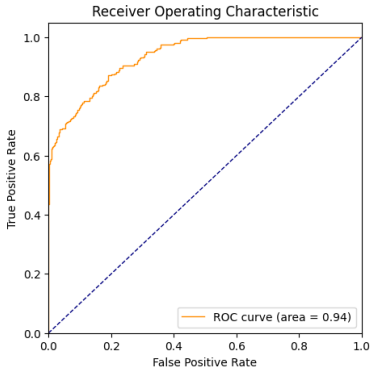


Figure 8: ROC curve

2. **Model 6**: This showcased high validation accuracy and F1 score, indicating effective identification of fake and real images. However, its lower Bitgrit score suggests a lesser generalization on unseen data. This model is an ensemble of MLP, Naive Bayes Classifier, KNN, and XGBoost.

## 4 Discussion

Model 1 and Model 6 both exhibited interesting performances. Despite its lower accuracy and F1 score, Model 1 outperformed Model 6 in terms of Bitgrit score. This suggests that Model 1 may be more capable of handling new, unseen data, a crucial capability in real-world applications. The stacking architecture used in Model 1 likely lends it the robustness to handle such data, assisted by the Gaussian Naive Bayes final estimator that reduces overfitting risk.

On the other hand, Model 6 excelled in traditional performance metrics like accuracy and F1 score but fell behind on the Bitgrit score. It's possible that Model 6 overfit the training data, explaining its excellent validation set performance but poorer performance on unseen data.

The discrepancy between the performances of the two models on different metrics highlights the importance of considering multiple evaluation measures. It also underscores the value of robustness and generalizability, areas where Model 1 seems to excel.

Improving Model 6 might involve incorporating strategies to enhance its generalizability and reduce overfitting, including exploring different model architectures or employing more advanced regularization techniques. The success of both models supports the use of ensemble and stacking methods in combating digital deception, inviting further research in this direction.

## 5 Conclusion

In conclusion, the study examined a range of models for deepfake image detection, evaluated on three criteria: validation set accuracy, validation set F1 score, and Bitgrit score. Two models emerged as significant performers, each excelling in different areas. Model 1, a stacked model, demonstrated superior robustness and generalization capabilities, as evidenced by its high Bitgrit score. This underlines the efficacy of ensemble and stacking methods in addressing the evolving challenge of deepfake detection, especially when dealing with unseen data, a key factor in real-world applications.

On the other hand, Model 6 showed excellent performance in traditional measures of accuracy and F1 score but lagged in terms of generalization. This raises potential overfitting concerns, and it emphasizes the need for balance between model accuracy and generalizability. It also suggests potential future directions, including exploring diverse model architectures or advanced regularization techniques. Overall, as deepfake technology continues to advance, the methods to detect and counter them need to evolve concurrently, integrating a multifaceted approach in model design and evaluation.

## References

Bigrit. Competition datasets, 2023. `https://bitgrit.net/competition/18?ref=mlcontests`.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.