

Steganalysis LSB Matching pada Grayscale Image menggunakan Machine Learning

Syifa Nurgaida Yutia

Teknologi Informasi, Institut Teknologi Telkom Jakarta
Jalan Daan Mogot KM.11, Jakarta 11710 Indonesia
syifanurgaida@ittelkom-jkt.ac.id

Abstract

Pada era digital saat ini teknik-teknik menyembunyikan pesan atau steganografi pada sebuah data telah mengalami perkembangan. Sehingga diperlukannya pengembangan teknik untuk mendeteksi pesan tersebut yang dapat dilakukan dengan metode steganalisis. Tujuan steganalisis yaitu untuk mengetahui apakah dalam suatu objek berisi pesan rahasia atau tidak. Terdapat dua metode steganografi secara garis besar yaitu *LSB Replacement* dan *LSB Matching*. Penerapan *LSB Replacement* dalam steganografi menyebabkan ketidakseimbangan dalam histogram gambar yang dapat dengan mudah dieksploitasi dengan metode statistik untuk steganalisis. Metode kedua yaitu *LSB Matching* dapat memecahkan masalah pada *LSB Replacement* dengan secara acak menambah atau mengurangi nilai piksel dengan 1 dalam kasus ketidakcocokan bit LSB untuk menghindari masalah ketidakseimbangan histogram dan membuatnya sulit untuk melakukan steganalisis dengan metode statistik saja, maka untuk mengatasi hal tersebut pada penelitian ini melakukan teknik *steganalysis* dengan penggunaan *machine learning* untuk keberadaan *LSB matching* dalam gambar *grayscale* (skala abu-abu) menggunakan teknik klasifikasi.

Keywords: *steganography, machine learning, steganalysis, artificial intelligence, LSB.*

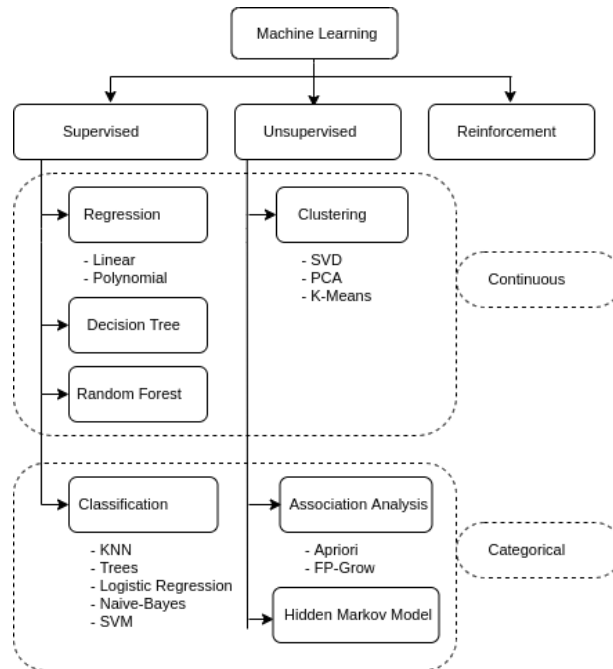
I. PENDAHULUAN

Steganalysis adalah ilmu untuk mendeteksi keberadaan pesan tersembunyi dalam suatu gambar, audio, video ataupun file yang tertanam dengan *steganography*. Algoritma *steganalysis* dikembangkan untuk menentukan apakah suatu gambar telah tertanam berdasarkan fitur statistik dari distorsi [1]. Berbagai metode *steganalysis* dapat dibagi menjadi *steganalysis* khusus dan *steganalysis* umum. Kategori pertama adalah metode *steganalysis* khusus yang biasanya memiliki tingkat deteksi lebih tinggi. Kategori kedua adalah deteksi awam yang tidak tergantung pada algoritma *steganography* khusus [2]. Dengan menggunakan teknik *steganography*, mitra yang berkomunikasi membangun saluran aman dalam sinyal *host* untuk mengirimkan pesan rahasia yang dianggap tidak terlihat oleh pihak ketiga. Saat ini, sebagian besar metode *steganography* pada gambar mencapai tujuan dengan memasukkan informasi rahasia ke dalam gambar asal yang tak terhindarkan meninggalkan bukti distorsi. Teknik *steganography* yang biasa digunakan yaitu teknik LSB (*Least Significant Bit*). Berdasarkan hal tersebut maka kebutuhan perkembangan teknik baru untuk alat *steganalysis* akan semakin tinggi yang dapat mendeteksi objek penutup atau *steganogram*. Dalam penelitian ini mengusulkan *Machine learning* untuk *steganalysis* dalam mendeteksi gambar *steganography* hasil metode dari *LSB Matching*. Penggunaan *machine learning* digunakan untuk mengajarkan mesin bagaimana menangani data dengan lebih efisien sehingga dapat menafsirkan pola atau mengekstrak informasi dari data. Bagian makalah ini disusun sebagai berikut: Bagian II menyajikan Tinjauan Pustaka. Pada bagian III Metodologi Penelitian, Bagian IV Hasil dan pengujian, dan akhirnya pada bagian V yaitu berisi kesimpulan.

II. LANDASAN TEORI

A. Machine Learning

Machine Learning merupakan teknik pembelajaran yang diterapkan pada mesin sehingga mesin bisa meniru kecerdasan yang dimiliki oleh manusia. *Machine Learning* dapat digunakan untuk melakukan pendeteksian, melakukan pengelompokan atau klasifikasi, dan mengidentifikasi kelas atau kategori dari suatu individu. Ada dua jenis teknik dalam *machine learning* yaitu *supervised* (pembelajaran terbimbing) dan *unsupervised* (pembelajaran tak terbimbing).



Gambar 1. Tipe *Machine Learning* dan Algoritma yang digunakan

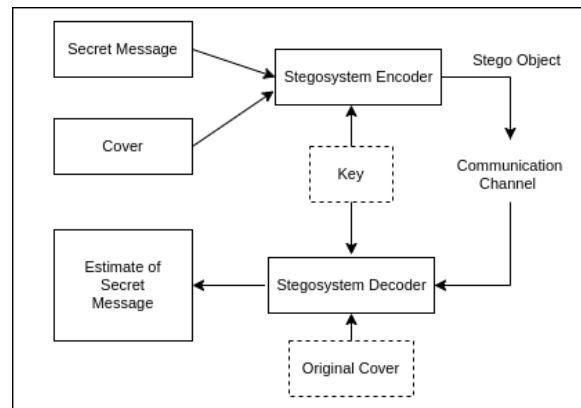
Supervised Learning merupakan teknik yang melatih model input dan output data yang diketahui sehingga dapat memprediksi output di masa depan, dan *Unsupervised Learning* merupakan teknik yang menemukan pola tersembunyi atau struktur intrinsik dalam data input [3].

B. Supervised Learning

Supervised Learning merupakan algoritma *machine learning* yang membutuhkan bantuan eksternal. Dataset input dibagi menjadi dataset *train* dan *test*. Dataset *train* memiliki variabel keluaran yang perlu diprediksi atau diklasifikasikan. Semua algoritma mempelajari semacam pola dari dataset *train* dan menerapkannya pada dataset uji/*test* untuk prediksi atau klasifikasi [3]. Tujuan *supervised machine learning* adalah untuk membangun model yang membuat prediksi berdasarkan bukti yang dihadapkan ketidakpastian.

C. Steganography

Steganografi adalah seni komunikasi terselubung yang berbeda dengan *cryptography*. Pada *Cryptography* bertujuan untuk membuat pesan rahasia tidak dapat dibaca, namun masih menarik perhatian dari penyadap karena jelas bahwa komunikasi dienkripsi. Sedangkan pada steganografi menyembunyikan pesan rahasia di dalam media sampul yang terlihat tidak mencurigakan dengan tujuan menyembunyikan keberadaan pesan dengan cara yang membuat komunikasi rahasia tidak terlihat [4] [5]. Jenis-jenis media pada penerapan *steganography* yaitu text, image dan audio/video. Gambar 2 menunjukkan model pada sistem steganografi.



Gambar 2. Model of steganographic systems

Sistem steganografi pada umumnya meliputi objek, pesan rahasia, *stego-key*, *stego-system encoder*, *stego object* dan *stego-system decoder*. Yang menyimpan informasi tersembunyi disebut sebagai objek Sampul. Kunci Stego adalah informasi tambahan, seperti kata sandi atau variabel matematika, yang diperlukan untuk menanamkan informasi rahasia. *Stego system encoder* merupakan proses penerapan kunci stego untuk menyembunyikan pesan rahasia di dalam objek penutup/*cover*. Kombinasi objek data dan *cover* tersembunyi dikenal sebagai objek stego. *Decoder* sistem Stego adalah proses penerapan kunci stego yang sama pada objek stego untuk memisahkan pesan tersembunyi dari objek penutup [6].

D. Steganography pada Gambar

Salah satu jenis media *steganography* yaitu gambar. Target *steganography* banyak memanfaatkan media gambar *digital*. Gambar *digital* memiliki nilai yang disebut elemen gambar atau piksel yang terdiri dari jumlah baris dan kolom piksel tetap. Dalam dunia komputasi digital, gambar digital dengan jenis *grayscale* adalah gambar di mana nilai setiap piksel membawa informasi intensitas, nuansa abu-abu, bervariasi dari hitam pada intensitas paling lemah hingga putih di terkuat. Saat ini, gambar *grayscale* yang ditujukan untuk tampilan visual umumnya disimpan dengan 8 bit per piksel. Dengan demikian, 256 jenis intensitas abu-abu digunakan. Gambar RGB (*Red, Green, and Blue*) memiliki tiga saluran: merah, hijau, dan biru sedangkan gambar skala abu-abu hanya memiliki satu saluran. Klasifikasi format pada gambar *digital* tergantung pada jumlah kolom yang membentuk gambar. Beberapa warna dalam suatu gambar juga mempengaruhi ukuran gambar. Jenis format gambar antara lain jpg, gif, tiff, png, bmp. Semua jenis file atau format ini digunakan untuk menyandikan gambar *digital*. Ada dua jenis kompresi untuk memampatkan gambar digital. Kompresi diterapkan menggunakan algoritma kompresi pada gambar digital. Algoritma kompresi *lossless* tidak membuat kompromi dengan keakuratan gambar *digital* karena informasi tersebut dibuang ke mana-mana sementara algoritma yang bertujuan kompresi *lossy* memperkenalkan beberapa degradasi gambar untuk mempersingkat ukuran *file*, karena mereka dapat menyimpan informasi warna pada resolusi lebih rendah daripada gambar itu sendiri. Di sisi lain untuk memotong ukuran *file* menggunakan algoritma kompresi *lossless* termasuk mengganti pola berulang *file* dengan semacam singkatan yang lebih pendek [6].

E. Steganalysis

Steganalisis merupakan teknologi untuk mendeteksi dan menganalisis file pembawa potensial untuk penyembunyian data menggunakan steganografi [7]. Tujuan dari steganalisis dapat tiga tingkatan termasuk: mendeteksi, mengekstraksi, dan menonaktifkan atau menghancurkan pesan tersembunyi [8].

F. Least Significant Bit (LSB)

Untuk mendeteksi keberadaan pesan tersembunyi sebagian besar dengan teknik mencari artefak yang dibuat oleh teknik penyembunyian pesan pada *steganography* yang biasa dilakukan dengan *LSB Replacement* dan *LSB Matching*. *LSB Replacement* hanya memengaruhi LSB dari setiap nilai data dengan menanamkan sebuah pesan ke dalam *cover image* dengan mengganti nilai LSB dari *cover image* dengan nilai bit pesan dan menghasilkan *stego image*. Metode ini menambahkan 1 pada nilai pixel genap dan mengurangi 1 pada nilai pixel ganjil atau tidak melakukan perubahan nilai sehingga menciptakan ketidakseimbangan dalam distorsi. Sejumlah teknik steganalisis memanfaatkan ketidakseimbangan ini. *LSB Matching*, di sisi lain memanfaatkan kenaikan atau penurunan nilai data ganjil dan genap dengan probabilitas yang sama yaitu dengan mengeksploitasi ketidakseimbangan [9]. *LSB matching* juga mengubah nilai LSB dari *cover image* dalam menyisipkan pesan. *LSB matching* tidak langsung mengubah nilai LSB dari *cover image* seperti yang dilakukan pada *LSB replacement*. Di mana jika bit pesan tidak cocok dengan LSB dari *cover image*, lalu nilai 1 secara acak akan ditambahkan atau dikurangi dari nilai pixel *cover*. Nilai pixel tidak pernah diubah menjadi nilai di luar batas yang diperbolehkan [10].

III. METODOLOGI PENELITIAN

Untuk strategi implementasi terdapat beberapa proses yaitu kebutuhan/*requirement*, analysis and *methodology*.

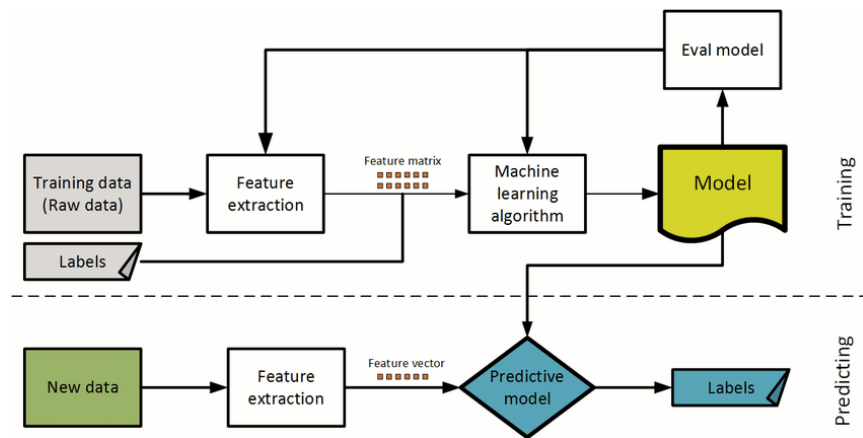
A. Kebutuhan/*Requirement*

Pada perancangan *Supervised Machine Learning* untuk *image steganalysis* meliputi persiapan kebutuhan/*requirement hardware* maupun *software* yang dijelaskan pada tabel 1.

TABLE I
KEBUTUHAN HARDWARE DAN SOFTWARE

<i>Machine Learning untuk Image Steganalysis</i>	
<i>Hardware</i>	• Acer Intel Core I3 • RAM 4 GB • Intel HD Graphics
Sistem Operasi	• Ubuntu 18.04 LTS (Bionic Beaver)
<i>Software Aplikasi</i>	• Jupyter Notebook • Tensorflow • Aletheia
Bahasa Pemrograman	Python
<i>DataSet</i>	• BossBase Greyscale Images • Bossbase LSB Matching

Setelah menentukan kebutuhan sistem, selanjutnya yaitu merancang *flowchart*. *Flowchart* yang akan digunakan yaitu *supervised machine learning* yang dijelaskan pada gambar 3 yang akan digunakan untuk mendeteksi *image steganalysis* pada makalah ini.



Gambar 3. Flowchart Diagram Supervised Machine Learning

Pada gambar 3 menjelaskan *flowchart* dari *supervised machine learning*. Pada langkah pertama, *Raw Data Training* dibersihkan (*Data Cleaning*) dan dimasukkan ke dalam ekstraksi fitur untuk memilih hanya fitur yang berguna dari data yang tersedia. Selanjutnya, label atau kelas yang dikenal dan fitur yang diekstraksi kemudian diteruskan ke fase pelatihan di mana algoritma *machine learning* digunakan untuk mengidentifikasi model yang baik yang memetakan input ke output yang diinginkan. Fase evaluasi memberikan umpan balik ke fase ekstraksi fitur dan pembelajaran agar penyesuaian ditingkatkan akurasi model. Proses pelatihan diulangi hingga tingkat akurasi yang diinginkan tercapai. Setelah model dibangun, itu adalah kemudian digunakan untuk memprediksi label data baru.

Metrik yang digunakan adalah *F-Score* sebagai metrik evaluasi untuk model benchmark, solusi dan ukuran akurasi. Pada *F-Score* *precision* p dan *recall* r dari tes untuk menghitung skor. p adalah jumlah hasil positif yang benar dibagi dengan jumlah semua hasil positif yang dikembalikan oleh pengklasifikasi, dan *recall* r adalah jumlah hasil positif yang benar dibagi dengan jumlah semua sampel yang relevan (semua sampel yang seharusnya diidentifikasi sebagai positif). *F-Score* mencapai nilai terbaiknya pada 1 (presisi dan *recall* sempurna) dan terburuk pada 0. Metrik di atas diformalkan sebagai berikut:

$$\begin{aligned}
 Precision &= TP / (TP + FP) \\
 Recall &= TP / (TP + FN) \\
 F-Score &= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}
 \end{aligned}$$

F-Score adalah metrik yang baik untuk masalah karena aplikasi steganalisis membutuhkan ketepatan dan daya ingat yang baik.

B. Analisa dan Implementasi

Pada tahap analisis yaitu melakukan eksplorasi data. Dengan menganalisis objek *stego image* yang diterapkan dengan teknik *LSB Matching*. Berbeda dengan teknik *LSB* yang lain, ketidakseimbangan yang disebabkan oleh *LSB Matching* dalam domain spasial tidak cukup jelas untuk dieksploitasi oleh teknik statistik saja. Set fitur CF terdiri dari 41 fitur dan didasarkan pada premis yang memberikan informasi spasial yang cukup untuk dilatih [11].

1) Data Collection

Langkah pertama yang dilakukan adalah memasukan data set pada tahapan data *collection* dari dua file csv ke dalam satu frame data, diikuti oleh penambahan kolom target dengan *coding* yang diperlihatkan pada gambar 4.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# Import both the csv files as data frames
data_cov = pd.read_csv("datasets/steg_features.csv", header = None)
data_steg = pd.read_csv("datasets/steg_lsb_features.csv", header = None)
```

Gambar 4. Import datasets

Sehingga menghasilkan baris data cover pertama pada gambar 4

data_cov.head()										
	0	1	2	3	4	5	6	7	8	9 ...
0	-0.317327	0.827515	0.760605	0.740966	0.721418	0.910647	0.861356	0.835196	0.815543	0.818339 ...
2	-0.503111	0.862970	0.802899	0.775813	0.751000	0.927452	0.889261	0.866067	0.848226	0.855546 ...
3	-0.182988	0.887022	0.835196	0.813357	0.789932	0.911072	0.861291	0.824739	0.795830	0.856713 ...
4	0.006107	0.932943	0.906990	0.897635	0.886993	0.970490	0.954652	0.944758	0.934639	0.934616 ...
5	-0.062837	0.842912	0.754166	0.697135	0.650435	0.856007	0.775904	0.732342	0.700388	0.774753 ...

Gambar 5. data cover head

Sehingga menghasilkan baris data stego pertama pada gambar 5

data_steg.head()										
	0	1	2	3	4	5	6	7	8	9 ...
0	-0.317249	0.827004	0.760402	0.740601	0.721245	0.909903	0.861059	0.834736	0.815029	0.817973 ...
2	-0.503030	0.862545	0.802756	0.775952	0.750876	0.927202	0.888695	0.865893	0.847715	0.855019 ...
3	-0.182986	0.885745	0.833919	0.812366	0.788900	0.909339	0.859644	0.823534	0.794631	0.855730 ...
4	0.005779	0.932725	0.907008	0.897505	0.886902	0.970202	0.954428	0.944286	0.934372	0.934324 ...
5	-0.062954	0.842023	0.753467	0.696795	0.650121	0.855838	0.775512	0.732004	0.700093	0.774611 ...

5 rows x 41 columns

Gambar 6. data stego head

2) Data Pre-Processing

Pada tahapan data preprocessing yaitu melakukan data *cleaning*, yang melibatkan penghapusan semua baris yang memiliki nilai nan pada gambar 7

```
# Drop rows with NaN values
data_cov = data_cov.dropna(axis = 0)
data_steg = data_steg.dropna(axis = 0)
```

Gambar 7. code data cleaning

Hasil data *cleaning* menghasilkan data *cov head* dan data *seg head* yang baru pada gambar 8 dan 9

In [11]: data_cov.head()										
Out[11]:										
	0	1	2	3	4	5	6	7	8	9 ...
0	-0.317327	0.827515	0.760605	0.740966	0.721418	0.910647	0.861356	0.835196	0.815543	0.818339 ...
2	-0.503111	0.862970	0.802899	0.775813	0.751000	0.927452	0.889261	0.866067	0.848226	0.855546 ...
3	-0.182988	0.887022	0.835196	0.813357	0.789932	0.911072	0.861291	0.824739	0.795830	0.856713 ...
4	0.006107	0.932943	0.906990	0.897635	0.886993	0.970490	0.954652	0.944758	0.934639	0.934616 ...
5	-0.062837	0.842912	0.754166	0.697135	0.650435	0.856007	0.775904	0.732342	0.700388	0.774753 ...

5 rows x 41 columns

Gambar 8. Data Cov head setelah *cleaning* data

```
In [12]: data_steg.head()
Out[12]:
```

	0	1	2	3	4	5	6	7	8	9	...
0	-0.317249	0.827004	0.760402	0.740601	0.721245	0.909903	0.861059	0.834736	0.815029	0.817973	...
2	-0.503030	0.862545	0.802756	0.775952	0.750876	0.927202	0.888695	0.865893	0.847715	0.855019	...
3	-0.182986	0.885745	0.833919	0.812366	0.788900	0.909339	0.859644	0.823534	0.794631	0.855730	...
4	0.005779	0.932725	0.907008	0.897505	0.886902	0.970202	0.954428	0.944286	0.934372	0.934324	...
5	-0.062954	0.842023	0.753467	0.696795	0.650121	0.855838	0.775512	0.732004	0.700093	0.774611	...

5 rows × 41 columns

Gambar 9. Data *StegHead* setelah *cleaning* data

Kemudian menghitung dan menampilkan jumlah baris dari data *cov* dan data *steg* pada gambar 10

```
print("Number of rows in data_cov:", len(data_cov))
print("Number of rows in data_steg:", len(data_steg))

Number of rows in data_cov: 9679
Number of rows in data_steg: 9679
```

Gambar 10. Jumlah baris data *cover* dan data *stego*

Mengambil 10 *variance* dengan melakukan *Principal Component Analysis*

```
from sklearn.decomposition import PCA

# Perform PCA and keep only the first 10 dimensions
pca = PCA(10)
pca.fit(good_X)

reduced_X = pca.transform(good_X)
reduced_X = pd.DataFrame(reduced_X)
reduced_X.head()
```

Gambar 11. Proses PCA

Menghasilkan 10 *variance* pada gambar 12

Explained Variance	
0	0.5748
1	0.2337
2	0.1093
3	0.0230
4	0.0166
5	0.0143
6	0.0075
7	0.0064
8	0.0039
9	0.0028

Gambar 12. *Explained Variance*

E. Feature Extraction

Pada *feature extraction* yaitu proses mengambil dataset dan label terkait sebagai input dan membaginya menjadi set *train* dan *test* dengan proses *cross validation* pada gambar 13

```
from sklearn.cross_validation import train_test_split

# Takes a dataset and corresponding label as input and splits it into train and test sets
def getCrossValidationDataSplit(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

    print("Training set has {} samples.".format(X_train.shape[0]))
    print("Testing set has {} samples.".format(X_test.shape[0]))

    return [X_train, X_test, y_train, y_test]
```

Gambar 13. *Cross Validation*

Kemudian proses Inisialisasi lima pengklasifikasi pada gambar 14

```
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

# Initialize five classifiers and sample sizes and return combined results for all possible classifier-sample-size pair
def getClassificationResults(X_train, y_train, X_test, y_test):
    clf_A = GaussianNB()
    clf_B = RandomForestClassifier()
    clf_C = AdaBoostClassifier()
    clf_D = SVC()
    clf_E = MLPClassifier()

    samples_100 = len(y_train)
    samples_10 = int(0.1 * samples_100)
    samples_1 = int(0.01 * samples_100)

    results = {}
    for clf in [clf_A, clf_B, clf_C, clf_D, clf_E]:
        clf_name = clf.__class__.__name__
        results[clf_name] = {}
        for i, samples in enumerate([samples_1, samples_10, samples_100]):
            results[clf_name][i] = trainPredict(clf, samples, X_train, y_train, X_test, y_test)

    return results
```

Gambar 14. *Initialize Five Classifiers*

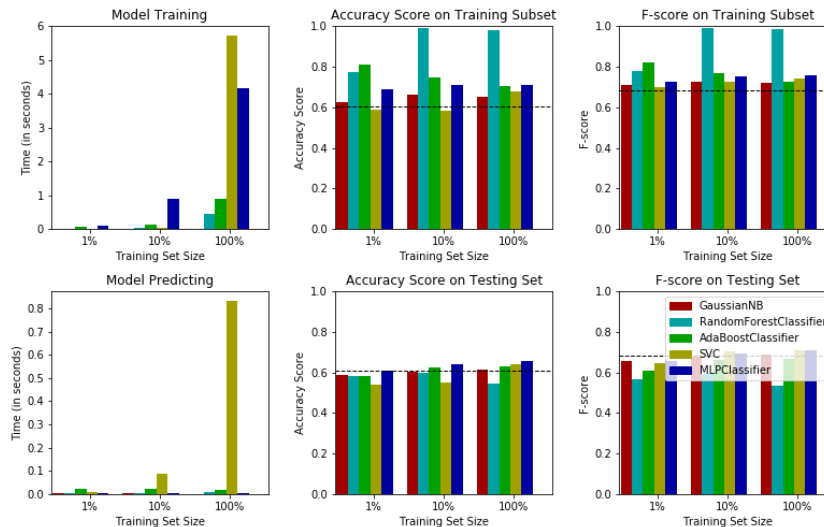
Melakukan proses *train* pada *original* dataset dan visualisasi hasilnya pada gambar 15

```
Training set has 15486 samples.
Testing set has 3872 samples.
Naive Predictor: [Accuracy score: 0.6059, F-score: 0.6819]
GaussianNB trained on 154 samples.
GaussianNB trained on 1548 samples.
GaussianNB trained on 15486 samples.
RandomForestClassifier trained on 154 samples.
RandomForestClassifier trained on 1548 samples.
RandomForestClassifier trained on 15486 samples.
AdaBoostClassifier trained on 154 samples.
AdaBoostClassifier trained on 1548 samples.
AdaBoostClassifier trained on 15486 samples.
SVC trained on 154 samples.
SVC trained on 1548 samples.
SVC trained on 15486 samples.
MLPClassifier trained on 154 samples.
MLPClassifier trained on 1548 samples.
MLPClassifier trained on 15486 samples.
```

Gambar 15. Proses train *original* dataset pada *classifier*

IV. HASIL DAN PEMBAHASAN

Setelah menerapkan pengklasifikasi pada masing-masing set *original* dataset dengan 5 *classifier* yaitu Gaussian Naïve Bayes, Random Forests, Support Vector Classifier, AdaBoost dan Multi-Layer Neural Network Perceptron. Untuk masing-masing pengklasifikasi, dicatat kinerja berdasarkan *train time*, *prediction time*, *train accuracy*, *test accuracy*, *train f-score* dan *test f-score* untuk ukuran *train* yang berbeda, dan membandingkan akurasi dan skor-f dengan skor dari Benchmark Gaussian Naïve Bayes Predictor pada dataset asli. Hasil menunjukkan bahwa semua pengklasifikasi kecuali MLP Classifier meningkatkan kinerja ketika *train* pada dataset yang dibersihkan dari dataset asli. F-Score Classifier MLP menurun sekitar 1%. Saat membandingkan waktu pelatihan, SVC membutuhkan waktu paling lama, diikuti oleh MLP classifier dan AdaBoost. Namun, dalam kasus ini untuk waktu *train* yang lama bisa ditolerir demi kinerja yang lebih baik. SVC juga memiliki waktu pengujian yang sangat tinggi. Ini membuatnya kurang optimal untuk karena dalam model akhir ekstraksi fitur CF itu sendiri membutuhkan waktu yang lama, dan menghemat waktu sebanyak mungkin penting dalam komponen prediksi. Untuk *train* dataset yang lebih sederhana termasuk Naive Bayes dan SVC juga bekerja dengan sangat baik, jika dibandingkan dengan model lainnya. Secara keseluruhan, bahwa MLP Classifier dan AdaBoost secara konsisten mengungguli semua pengklasifikasi lain dalam hal akurasi pengujian dan uji *f-score* terlihat pada gambar 16.



Gambar 16. Hasil Skor Akurasi

Berikut tabel untuk akurasi dari 5 *classifier*

	Model	Train Time	Training Accuracy	Training F-Score	Prediction Time	Test Accuracy	Test F-Score
0	GaussianNB	0.015673	0.580000	0.663102	0.002817	0.592717	0.670428
1	RandomForestClassifier	0.949791	0.983333	0.984227	0.008735	0.697314	0.684607
2	AdaBoostClassifier	3.733127	0.703333	0.750700	0.024033	0.707386	0.739000
3	SVC	19.401956	0.630000	0.694215	2.951462	0.636880	0.692341
4	MLPClassifier	5.667073	0.736667	0.783562	0.003764	0.720041	0.762801

Gambar 17. Tabel Akurasi

V. Kesimpulan

Machine Learning untuk kasus steganalysis dilakukan untuk mengatasi *image steganografi* dengan teknik *LSB matching* yang lebih sulit dideteksi dibandingkan teknik *image steganography* pada *LSB replacement*. *LSB Matching* adalah versi modifikasi dari *LSB Replacement*, dengan menggunakan dataset berlabel gambar skala abu-abu menggunakan *supervised* yaitu melalui proses data *cleansing*, data *pre-processing*, *feature* extraction, pengurangan dimensi melalui PCA dan proses *train* beberapa pengklasifikasi. Setelah menerapkan pengklasifikasi pada masing-masing set original dataset dengan 5 classifier yaitu Gaussian Naïve Bayes, Random Forests, Support Vector Classifier, AdaBoost dan Multi-Layer Neural Network Perceptron. Untuk masing-masing pengklasifikasi, dicatat kinerja berdasarkan train time, prediction time, train accuracy, test accuracy, train f-score dan test f-score. Dari hasil pengujian didapat *classifier* Multilayer Perceptron (MLP) yang terbaik dalam memodelkan keputusan yang sifatnya kompleks. Karena kinerjanya yang tinggi, MLP ideal untuk masalah *image steganalysis*. Namun untuk *classifier* yang lain juga memiliki kelebihan dan kekurangan masing-masing seperti untuk train dataset yang lebih sederhana termasuk Naive Bayes dan SVC juga bekerja dengan sangat baik, jika dibandingkan dengan model lainnya. Maka untuk pemilihan *classifier* tergantung dengan permasalahan atau kasus yang ingin diselesaikan.

REFERENCES

- [1] C. Cachin, "An information-theoretic model for steganography", in *Information Hiding*. Berlin, Germany: Springer, 1998, pp. 306–318. [Online]. Available: https://doi.org/10.1007/3-540-49380-8_21.
- [2] Z. Sun, H. Li, Z. Wu, and Z. Zhou, "An Image Steganalysis Method Based on Characteristic Function Moments of Wavelet Subbands, Artificial Intelligence and Computational Intelligence", AICI '09. International Conference on Volume: pp 291 - 295, 2009.
- [3] S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", *Informatica* 31 (2007) 249-268
- [4] R. J. Anderson and F. A. Petitcolas, "On the limits of steganography," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 4, pp. 474–481, 1998.
- [5] G. Huayong, H. Mingsheng, and W. Qian, "Steganography and steganalysis based on digital image," in *Image and Signal Processing (CISP), 2011 4th International Congress on*, vol. 1. IEEE, 2011, pp. 252–255.
- [6] R. Jain and J. Boaddh, "Advances in digital image steganography," *International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, 2016.
- [7] D. Artz, "Digital steganography: hiding data within data," *internet computing IEEE*, vol. 5, no. 3, pp. 75–80, 2001.
- [8] N. F. Johnson, S. Jajodia, "Steganalysis: The investigation of hidden information," *Information Technology Conference* 1998. IEEE, pp. 113–116, 1998.
- [9] T. Qian and S. Manoharan "A Comparative Review of Steganalysis Techniques," *International Conference on Information Science and Security (ICISS)*, 2015.
- [10] J. Mielikainen, "LSB Matching Revisited," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 285–287, 2006.
- [11] Q. Liu, A.H. Sung, J. Xu, and B.M. Ribeiro, "Image Complexity and Feature Extraction for Steganalysis of LSB Matching Steganography," *18th International Conference on Pattern Recognition (ICPR'06)*, 2006.