Detecting Midjourney Images Via Feature Engineering & Classification

Hairton Javier Sarmiento Yahuarcani

School of Engeneering. Universitat Rovira I Virigili

CF-L290: Computer Forensics

Fran Casino Cembellin

November 2nd, 2025

Introduction

Analyzing images to identify between real images from AI images is possible thanks to the forensics that can be done to images. Images have different features, which some are replicated by AI models, while others have a completely different structure due to the natural condition that only exist in the reality. These features are useful to differentiate real from AI images. Besides, ML and DL provide tools to make this identification optimal, and it is crucial to tune models and check how they are performing. Therefore, it is possible to improve models for this type of tasks.

Structure of the code

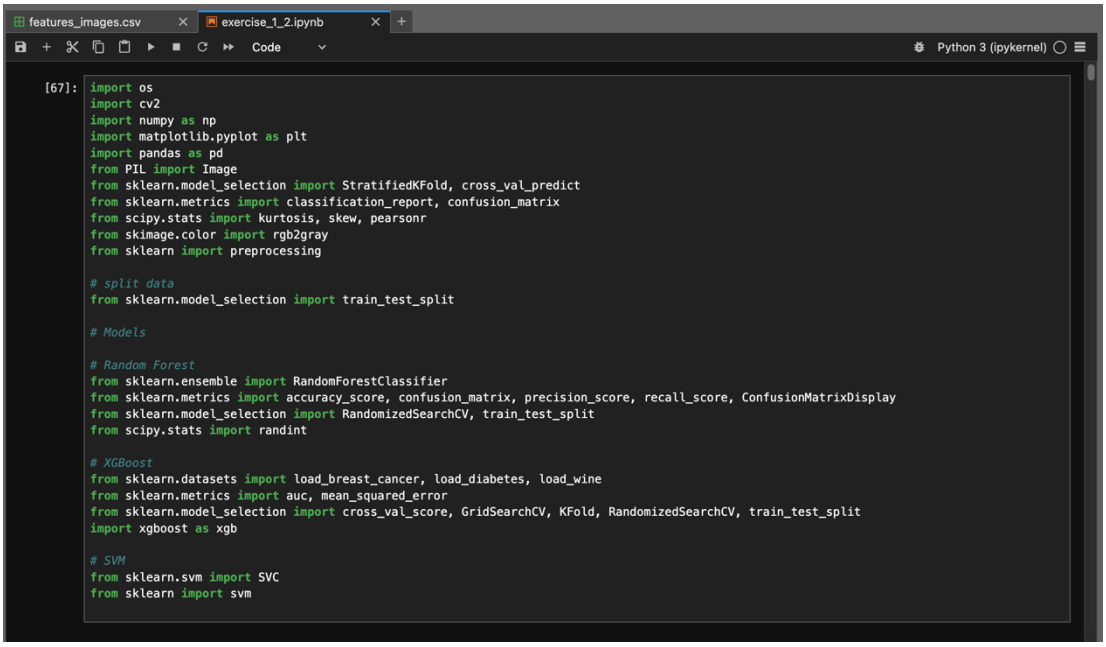
Along with this document, is attached a jupyter notebook that can be opened using either jupyter lab or VSC. Besides, the changes that were made in this file are logged in the following link, in case you would like to see how the code has evolved.

The code has the following cells:

Libraries: we have all the libraries needed to get the features and create the models.

Figure 1

Libraries



Features: After importing the needed libraries, you will find a markdown cell, indicating the feature that will be got and then the necessary code to get it. Besides, an example of how the feature is will be find in a third cell.

Figure 2

Feature cells

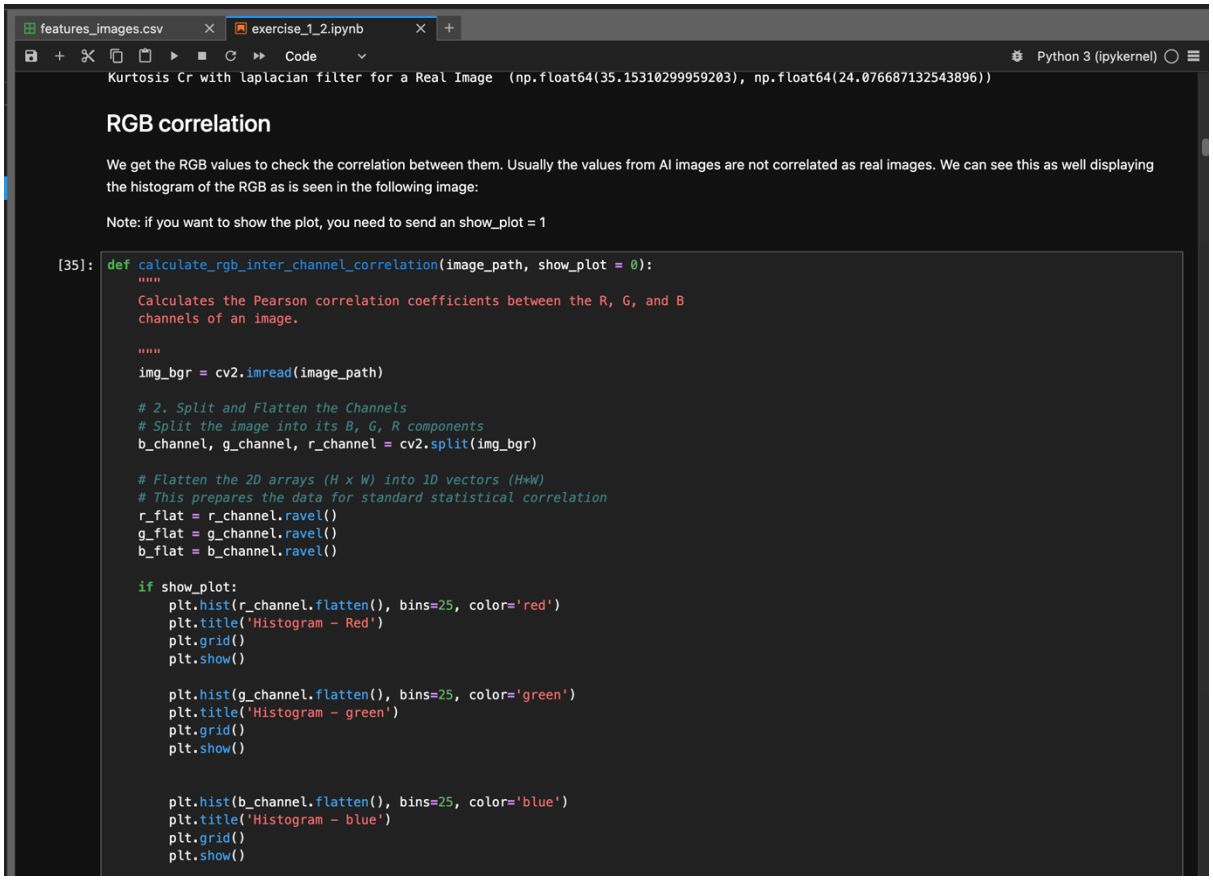
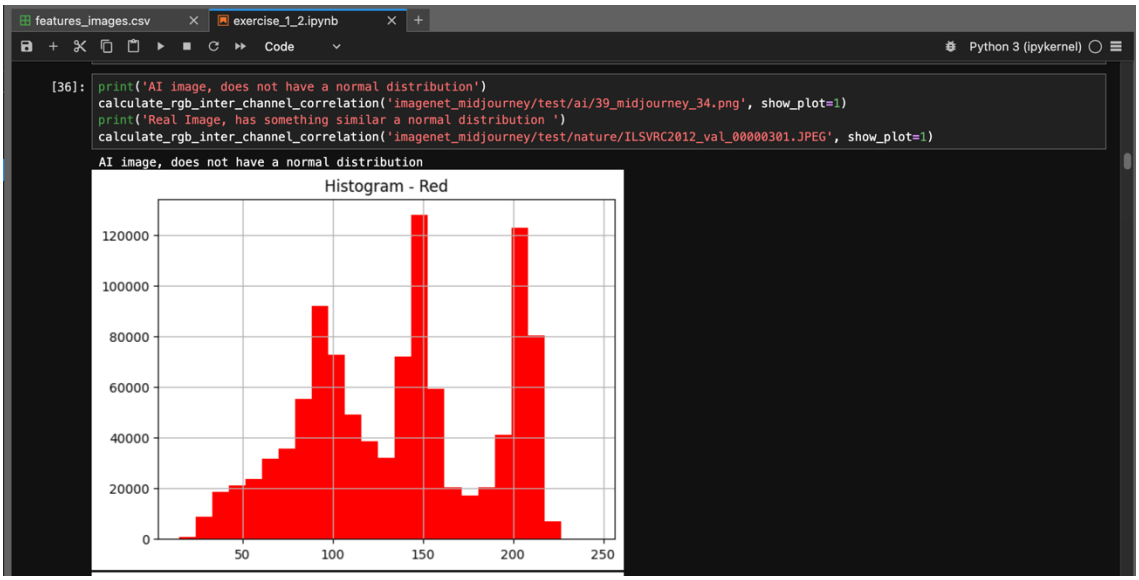


Figure 3

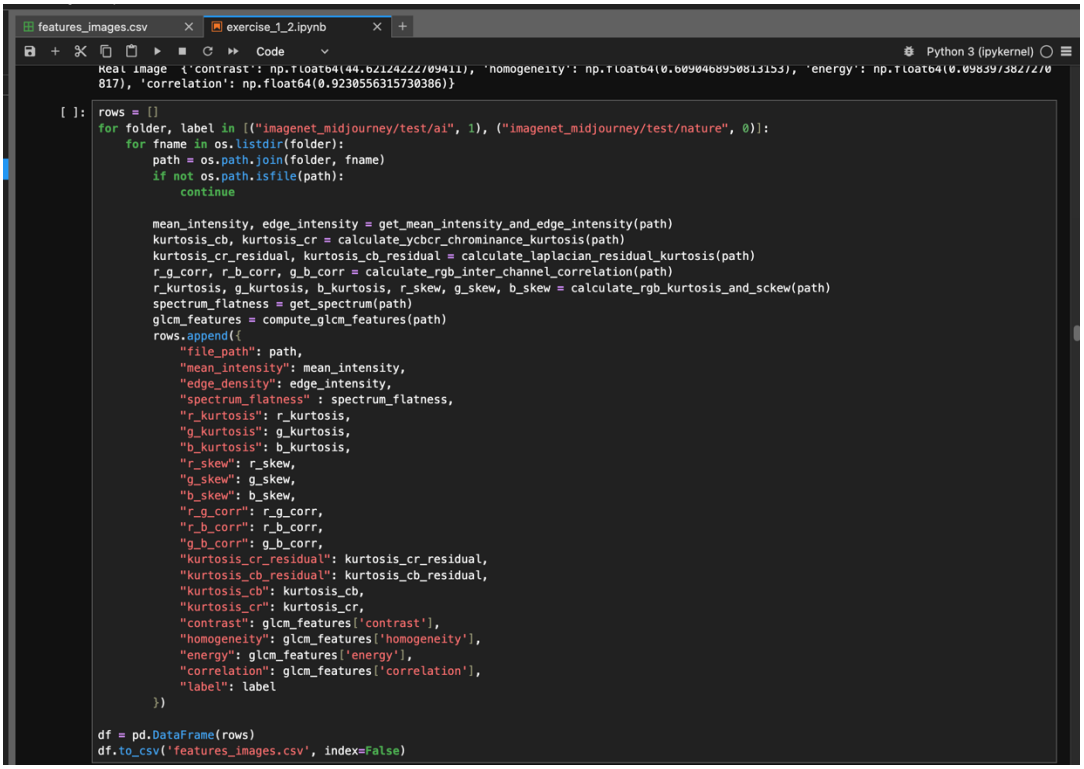
Example between real image and AI image using a feature



Scraper: After getting the 20 features. You will find the main function, where the df is made and saved in a CSV file.

Figure 4

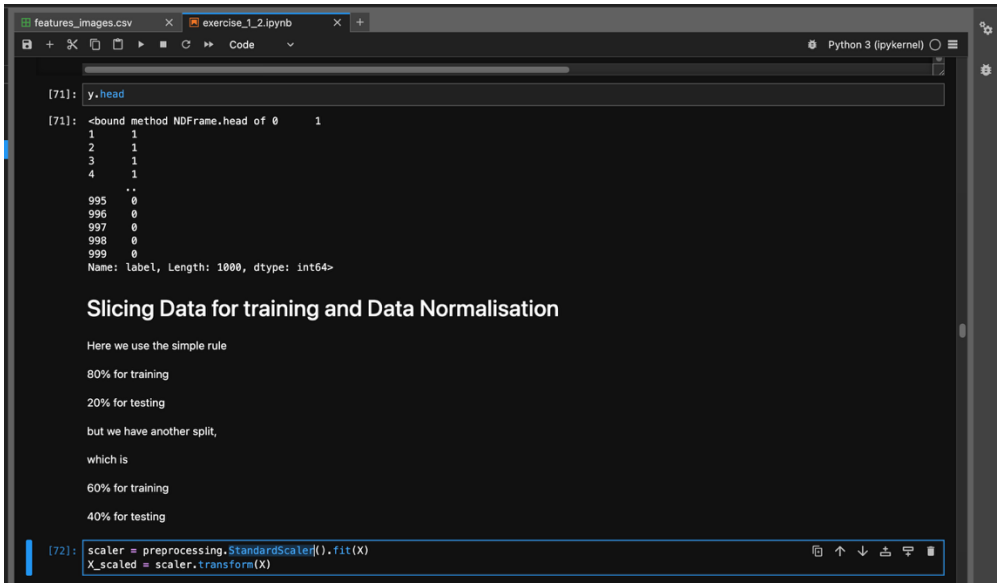
Main function to create data



Manage Data: After having the CSV or the df, the process of manipulating the data is done in the following cells. Basically, separating between features and labels, and using a feature from scikit learn that is called StandardScaler. This function helps us to regularize our data. Which means, all the data is in the same scale. This is good and the models will generalize better the data.

Figure 5

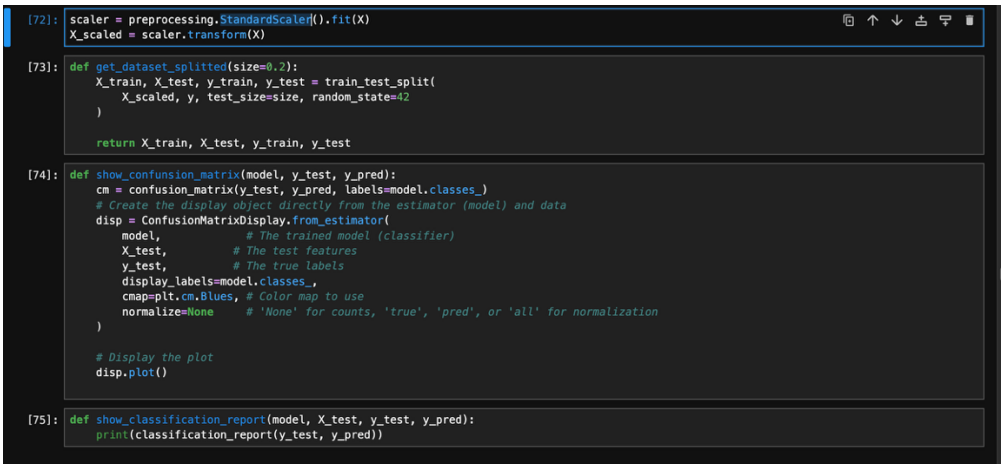
Pre-processed data



In this point, it is declared the helper functions that will be re-used with each model. For splitting the data, show the different metrics, show the confusion matrix and check the Gini and Permutation importance, which are useful to have intuition of the models used.

Figure 6

Helpers



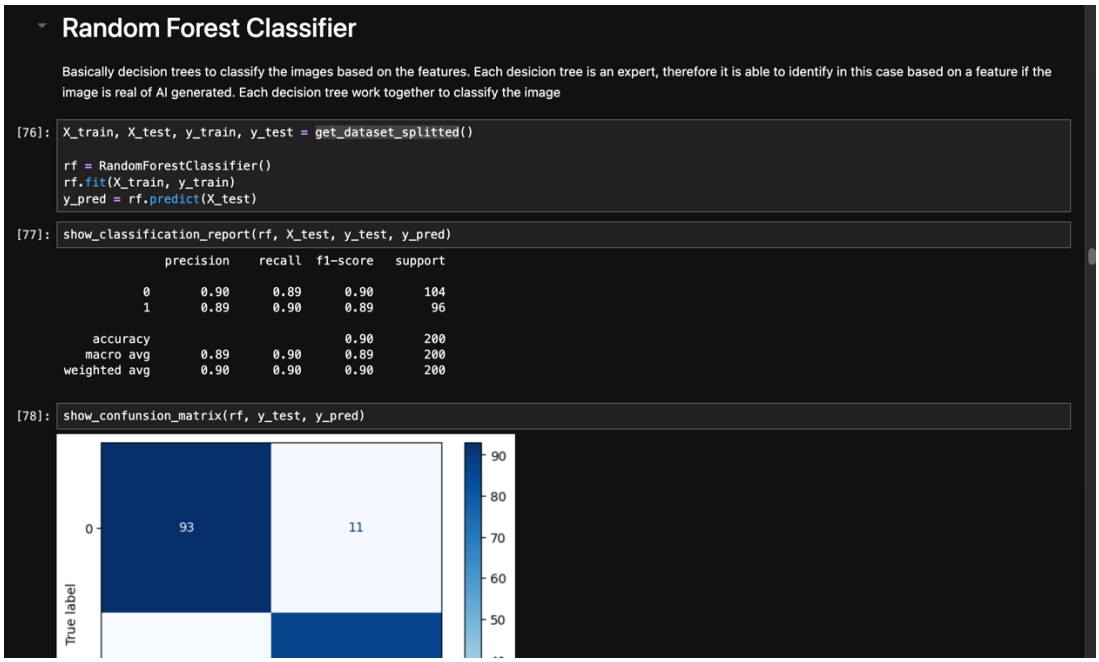
```
[72]: scaler = preprocessing.StandardScaler().fit(X)
      X_scaled = scaler.transform(X)

[73]: def get_dataset_splitted(size=0.2):
          X_train, X_test, y_train, y_test = train_test_split(
              X_scaled, y, test_size=size, random_state=42
          )

          return X_train, X_test, y_train, y_test

[74]: def show_confusion_matrix(model, y_test, y_pred):
          cm = confusion_matrix(y_test, y_pred, labels=model.classes_)
          # Create the display object directly from the estimator (model) and data
          disp = ConfusionMatrixDisplay.from_estimator(
              model,              # The trained model (classifier)
              X_test,           # The test features
              y_test,           # The true labels
              display_labels=model.classes_,
              cmap=plt.cm.Blues, # Color map to use
              normalize=None     # 'None' for counts, 'true', 'pred', or 'all' for normalization
          )

          # Display the plot
          disp.plot()

[75]: def show_classification_report(model, X_test, y_test, y_pred):
          print(classification_report(y_test, y_pred))
```

Models: The models used for this task were Random Forest, XGB and SVM using linear kernel. They are implemented in the following part. Each time a new model is used, a markdown cell will be displayed showing the model that is used. In each model, there were three implementations and one cell for showing feature importance:

- Model using a normal split 80/20, and no cross validation.
- Model using split 40/60 or 30/70 (only for SVM) and no cross validation
- Model using split 20/80 or 30/70 (only for SVM) and cross validation This way we can see how the models are performing when we tune the models.
- Cell for showing how the features are crucial for a model

Figure 7

Model Cell



Each model shows the metrics precision, recall, f1-score, support, and accuracy. Besides, a confusion matrix is shown too, which is helpful to understand better the classifications results.

Features

The following features were obtained:

YCbCr Chroma

Measures how color varies independently of brightness. The features are chrominance or better known Cb and Cr channels. In real images, the chroma behaves with natural chroma variations, smooth transitions, and realistic color dependencies. While AI images have unnatural or over-smoothed, exaggerated saturation, or shifted tones. Generators sometimes mismanage color consistency or chroma-noise balance, producing unnatural color gradients.

For this feature we get the channel and calculate kurtosis (Kenton, 2025). Therefore, we can identify an image by this statistical feature. Usually, AI images have lower value than real images.

Figure 8

Results YCbCr Chroma

```
Kurtosis Cb AI image  (np.float64(-0.36192062260120705), np.float64(-1.2480978607942883))
Kurtosis Cr Real Image  (np.float64(1.0340358001263619), np.float64(11.935706833326927))
```

Chroma Residuals

Residuals after applying Laplacian filter to the chroma channels. This filter highlights fine chromatic noise and high-frequency color details. Real images usually contain fine, stochastic color noise and camera sensor artifacts. While AI images are smoother and less random with weak high frequency chroma residuals because AI models often lack realistic sensor-level noise and color aliasing, giving smoother chroma residuals. Using csv2 we get a distribution and then the kurtosis to have a value, AI images often have lower scores than real images.

Figure 9

Results Chroma residuals

```
Kurtosis Cb with laplacian filter for an AI image  (np.float64(1.0884449770995488), np.float64(1.238487633165665))
Kurtosis Cr with laplacian filter for a Real Image  (np.float64(35.15310299959203), np.float64(24.076687132543896))
```

RGB Correlations

Measures the correlation between R, G, and B channels using the Pearson correlation. Real images have a moderate correlation due to natural lighting but with independent texture variations. However, AI images have too high or too low channels that may be overly correlated or decoupled. Often AI models may produce overly clean color dependencies, missing real sensor crosstalk or natural chromatic aberration. The best way to differentiate with this feature is using a plot but this feature gets the correlation between the channels.

Figure 10

Results RGB Correlations – histogram – IA image

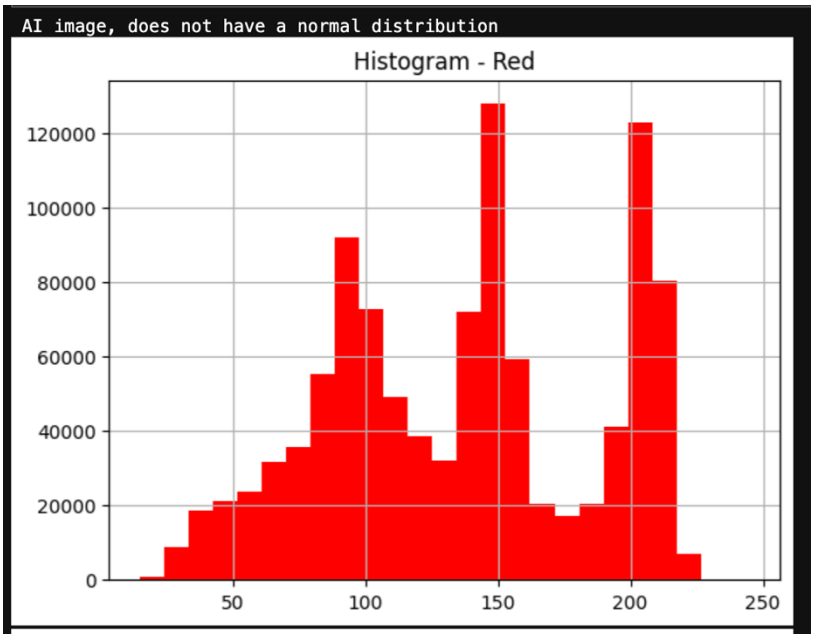
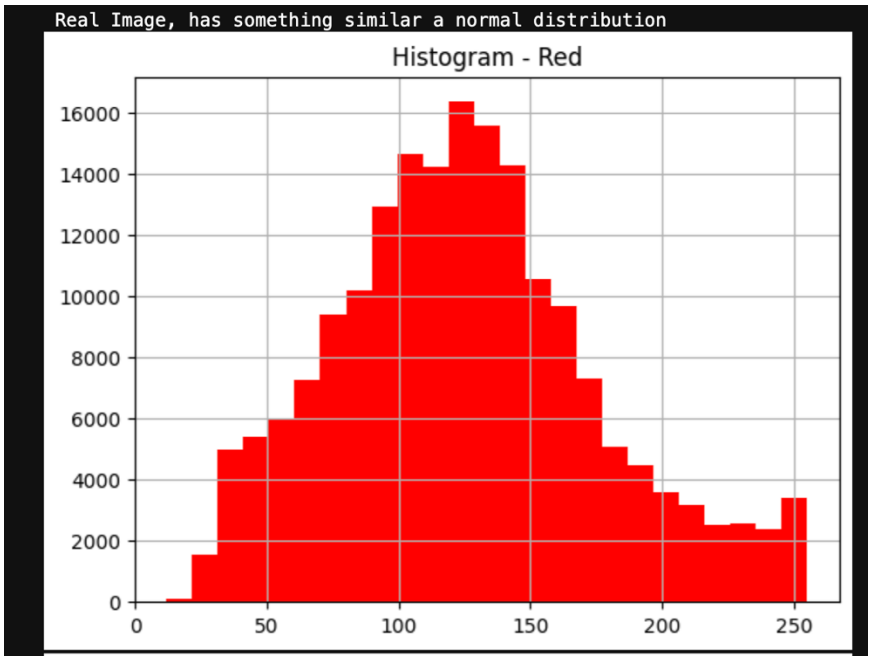


Figure 11

Results RGB Correlations – histogram – real image



Kurtosis and Skew of RGB

These are statistical moments of the RGB pixel distributions.

Kurtosis: peakiness / tail heaviness

Skew: symmetry of the distribution.

In real images, RGB histograms tend to have natural, slightly skewed, moderate kurtosis due to lighting variation. While AI images have kurtosis often lower (like in the images above) and skew more symmetric. AI images often have lower values than real images.

Figure 12

Results Kurtosis and Skew of RGB

```
AI image  (np.float64(-1.0282456003384026), np.float64(-0.6266833023005898), np.float64(-1.54413378330951), np.float64(-0.043085115159591944), np.float64(-0.5786945201539994), np.float64(-0.19364190365243364))
Real Image  (np.float64(-0.12849335875416257), np.float64(0.03938413144363828), np.float64(0.4945323390292664), np.float64(0.3863433175208686), np.float64(0.5131025227216862), np.float64(0.7170307401879993))
```

Spectral Flatness

It quantifies how noise-like the image's frequency spectrum is. Basically, the ratio of geometric to arithmetic mean of the power spectrum. Natural images have moderate flatness and a mixture of structured and random components. While AI images show flat spectrum or too peaked AI textures lack real camera frequency irregularities; can be **over-smooth** or **over-regular**, unlike natural frequency spectra. The best way to compare these two is showing the spectrum.

Figure 13

Results Spectrum and spectral flatness – real image

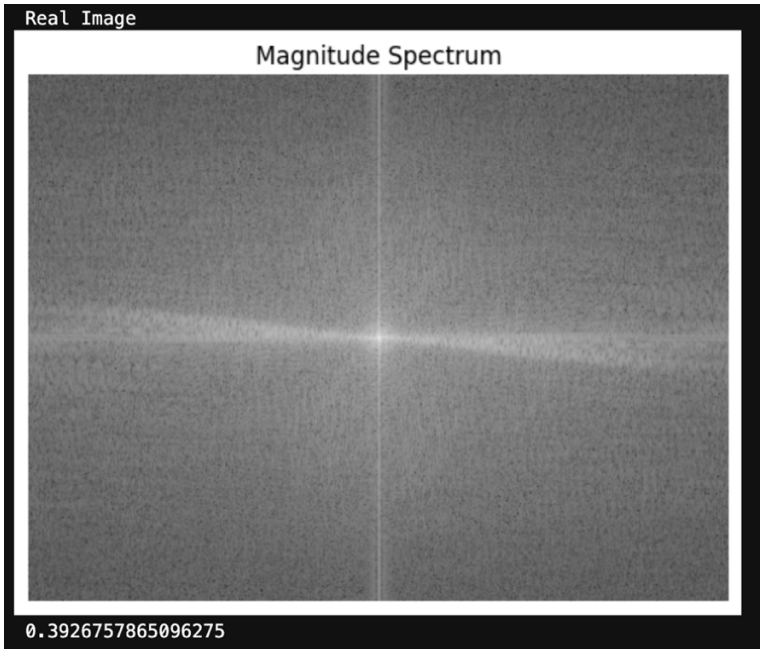
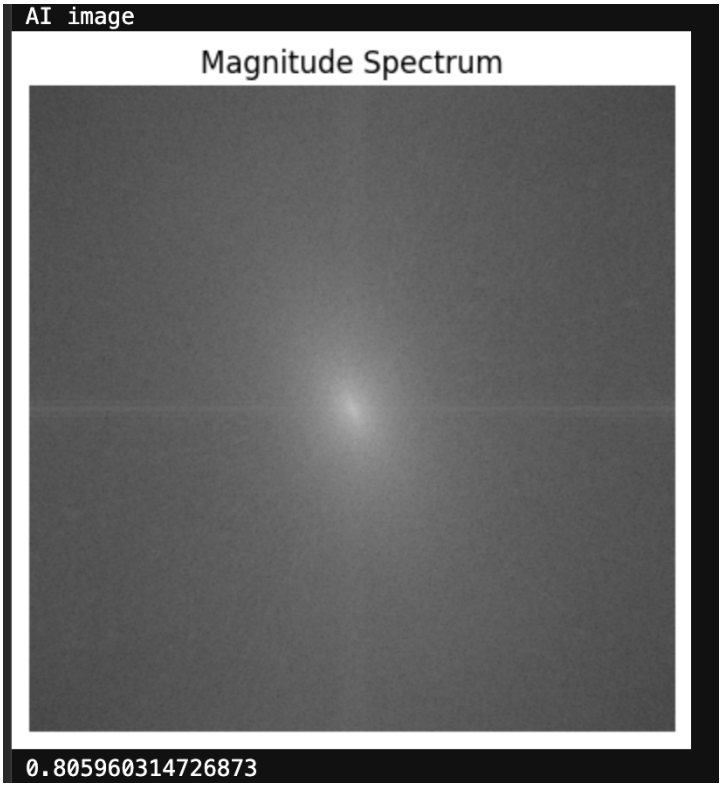


Figure 14

Results Spectrum and spectral flatness – AI image



Mean Intensity

It is the average brightness of the image. Real images are usually wide, scene-dependent range, and natural exposure balance. While AI images have limited range due to generator normalization. Generators typically normalize output luminance, leading to reduced exposure variability.

Edge Intensity

It is the average magnitude of gradients or edge map using via Sobel filter. Real images have a realistic distribution of strong and weak edges. While AI images have often either too smooth

(low edge intensity) or too sharp (artificial edges). AI models struggle with natural edge noise and texture boundaries, resulting in atypical edge statistics.

Figure 15

Results mean intensity and Edge Intensity

```
AI image  (121.8028736114502, 0.4563779830932617)
Real Image  (103.38722133333333, 0.13644266666666666)
```

Contrast

It measures intensity variation between neighboring pixels. Basically, high contrast means strong edges and therefore, abrupt changes. Real images are moderate, with natural edges and smooth transitions. While AI image has often lower or overly uniform, due to generative model smoothing or artificial sharpness. This value is higher on real images than in AI images.

Figure 16

Results contrast

```
AI image  {'contrast': np.float64(30.182466993042336),
774), 'correlation': np.float64(0.9190158651600533)}
Real Image  {'contrast': np.float64(44.62124222709411),
817), 'correlation': np.float64(0.9230556315730386)}
```

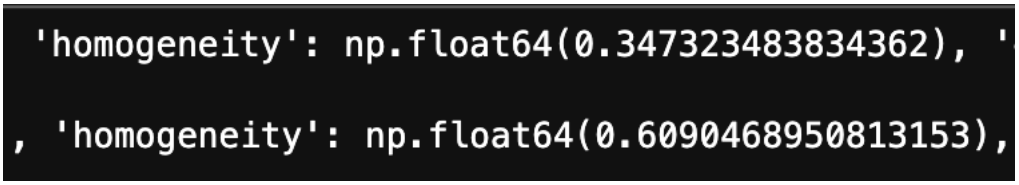
Homogeneity

It measures how uniform the texture is. How high when neighboring pixels have similar intensity. Real images vary naturally and edges reduce homogeneity. While AI images have

smooth and less varied micro-textures and synthetic textures lack chaotic local irregularities found in natural surfaces. This value is higher on real images than in AI images

Figure 17

Results homogeneity

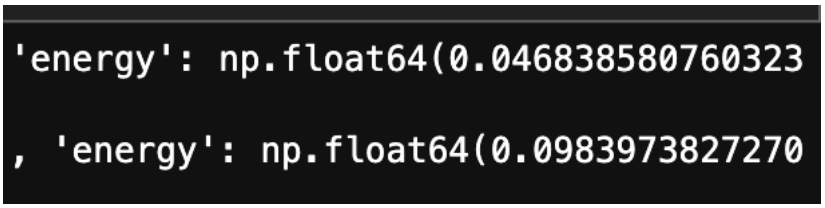


Energy

It measures textural uniformity; it is usually high when pixel pairs are concentrated in certain values. Real images have diversity of pixel relationships. While AI image often has **very high** due to repeated or over-regular patterns. Neural networks can overfit local texture patterns, leading to repetitive uniform structures. This value is higher on real images than in AI images

Figure 18

Results energy



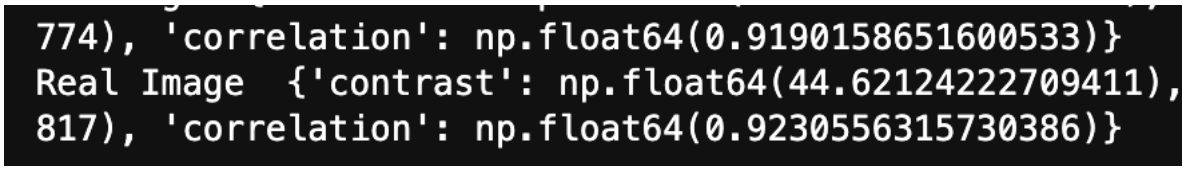
Correlation

It measures linear dependency between pixel pairs and repetitive pattern strength. Real images have lower values because real scenes mix textures and edges. While AI images have a higher

value since generative models produce globally correlated patterns. This should be different but checking with a few images, it is not a specific feature for comparing between real and AI images.

Figure 19

Results correlation



Models

A brief description of each model is useful to understand better the model used in this assignment.

Random Forest

Random Forest is a model that builds many independent decision trees on random subsets of your data and features, then based on the majority vote for a classification, gives a result (Shafi, 2025).

XGBoost

XGBoost is a model that builds trees sequentially. Each new tree learns from the errors of the previous ones and uses gradient descent to minimize a loss function, leading to highly accurate and efficient models (xgboosting, n.d.).

Support Vector Machine

SVM is a model that finds the optimal separating boundary (hyperplane**)** between classes that maximizes the margin or the distance between the closest points using support vectors of each class. For non-linear data, it uses kernel functions to project data into higher-dimensional space (GN, 2026).

Performance and Metrics

To illustrate in a better way the models used and the proposed experiment like data splitting, hyperparameter tunning, evaluation metrics and confusion matrix. The following table shows each model comparing the different features Precision, Recall, F1-score, Report, Split, Confusion Matrix and Accuracy. Besides, the table offers information about cross validation, hyperparameter tunning, and data splitting. It is important to mention that each model got a score of 85% of higher using the metric F1-score.

Table 1

Results of classification with Random Forest, XGB and SVM(Linear)

	Precision	Recall	F1-score	Report	Split	Accuracy	table	Cross Validation
Random Forest	0.90	0,91	0,90	200	80/20	0,91	Image 18	no
Random Forest	0,88	0,88	0,88	400	40/60	0,88	Image 19	no
Random Forest	0,92	0,91	0,91	200	80/20	0,92	Image 20	yes
XGB	0,89	0,90	0,89	200	80/20	0,90	Image 21	no
XGB	0,90	0,90	0.90	400	40/60	0,90	Image 22	No

XGB	0,91	0,92	0,91	200	80/20	0,92	Image 23	yes
SVM (linear)	0,84	0,84	0,83	200	80/20	0,83	Image 24	no
SVM (linear)	0,85	0,85	0,85	300	70/30	0,85	Image 25	no
SVM (linear),	0,85	0,85	0,85	300	70/30	0,85	Image 26	yes

Figure 20

Confusion Matrix using random Forest without cross validation and split 80/20

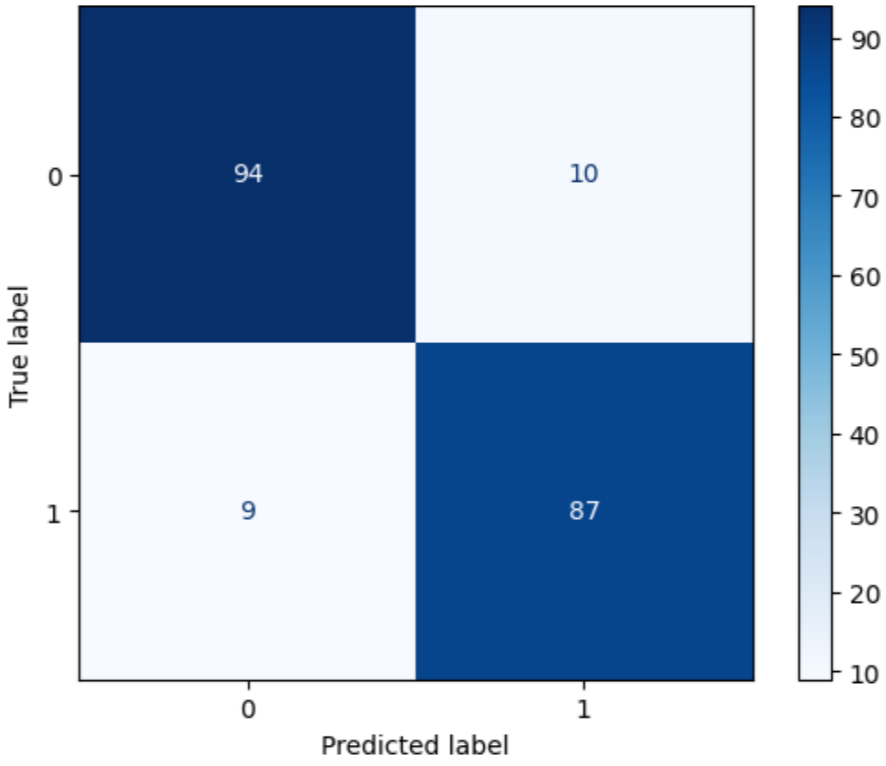


Figure 21

Confusion Matrix using random Forest without cross validation and split 60/40

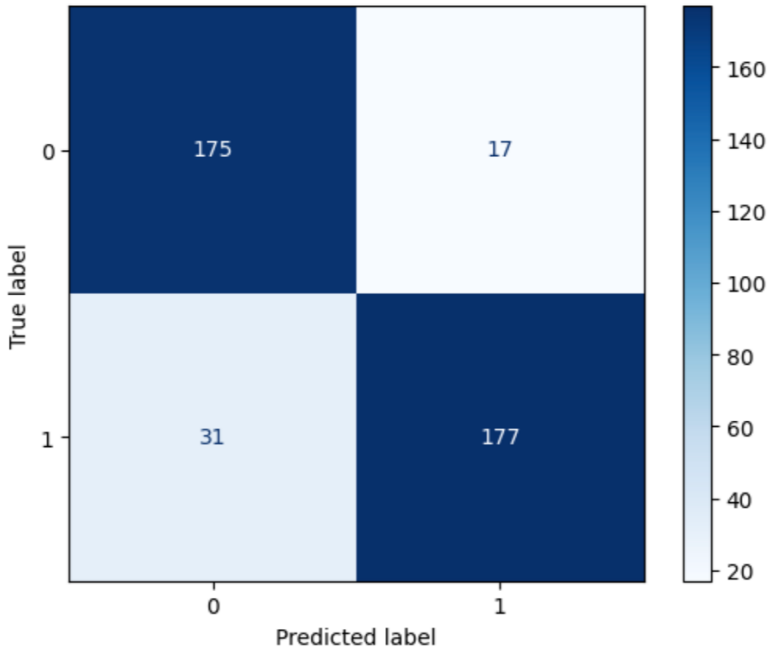


Figure 22

Confusion Matrix using random Forest with cross validation and split 80/20

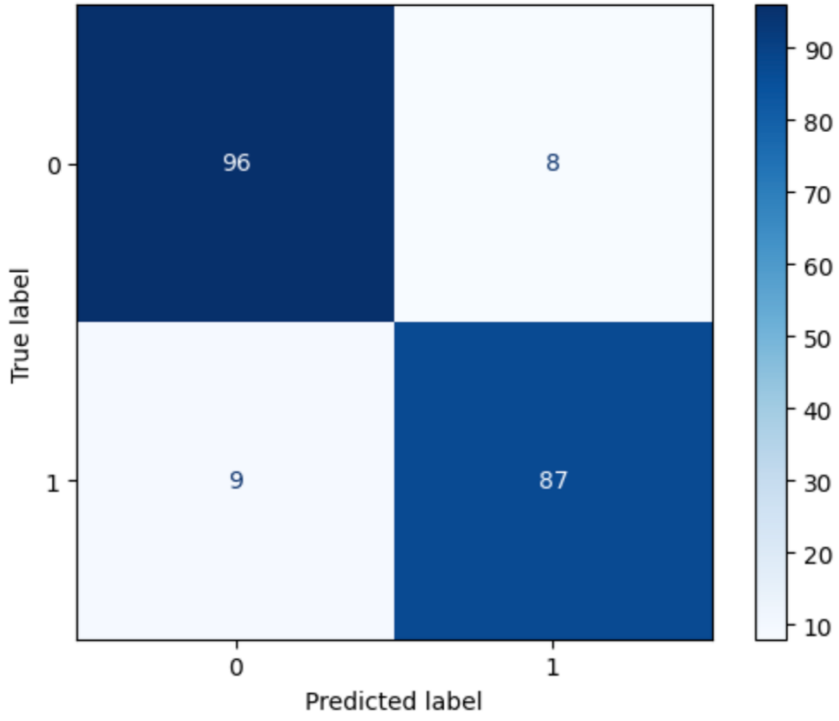


Figure 23

Confusion Matrix using XGB without cross validation and split 80/20

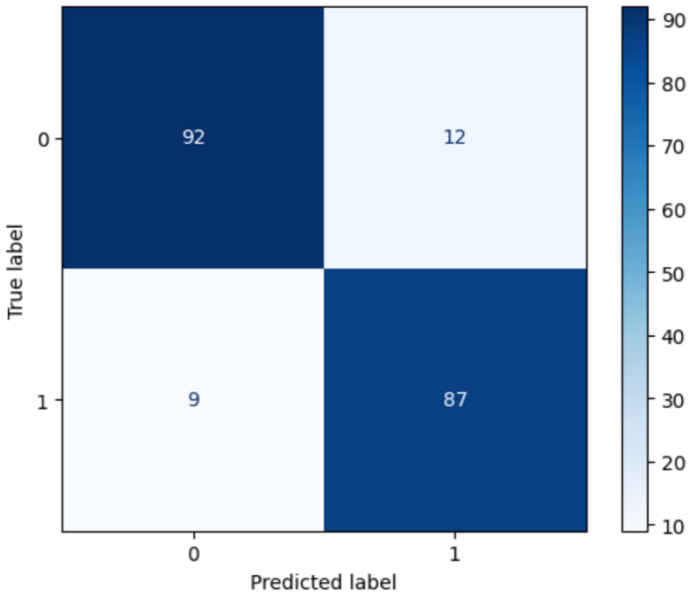


Figure 24

Confusion Matrix using XGB without cross validation and split 60/40

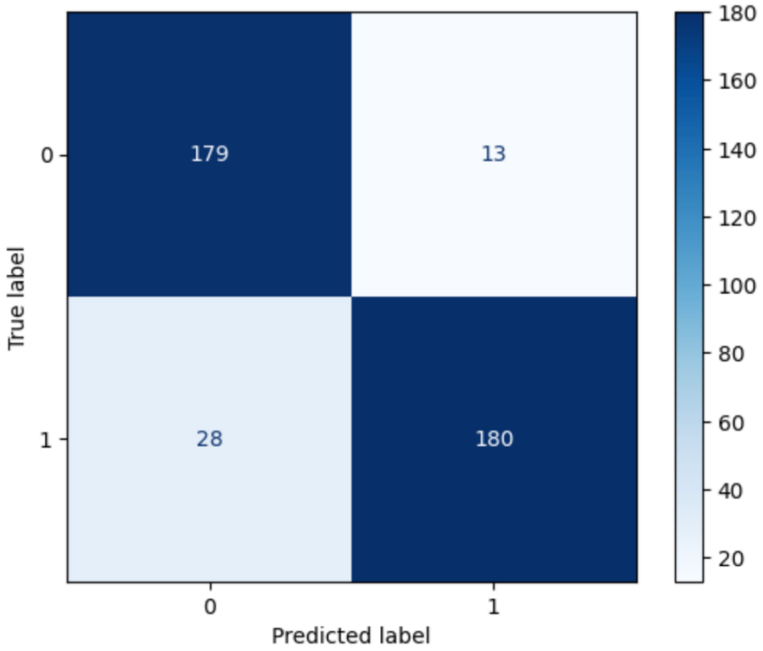


Figure 25

Confusion Matrix using XGB with cross validation and split 80/20

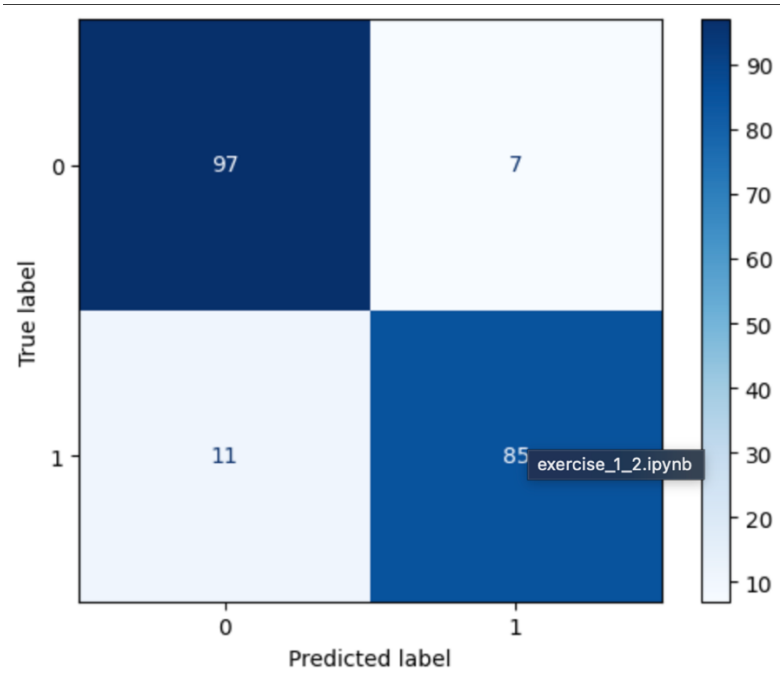


Figure 26

Confusion Matrix using SVM (linear) without cross validation and split 80/20

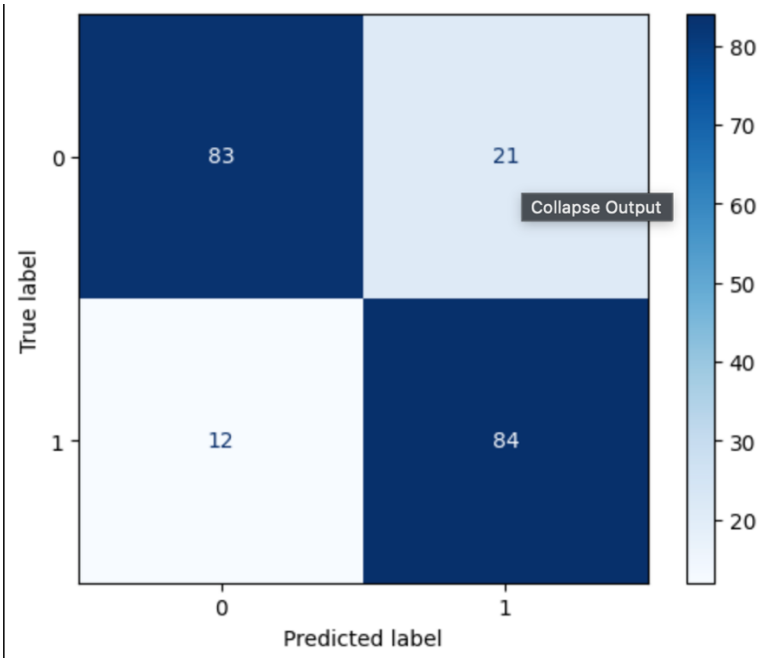


Figure 27

Confusion Matrix using SVM (linear) without cross validation and split 70/30

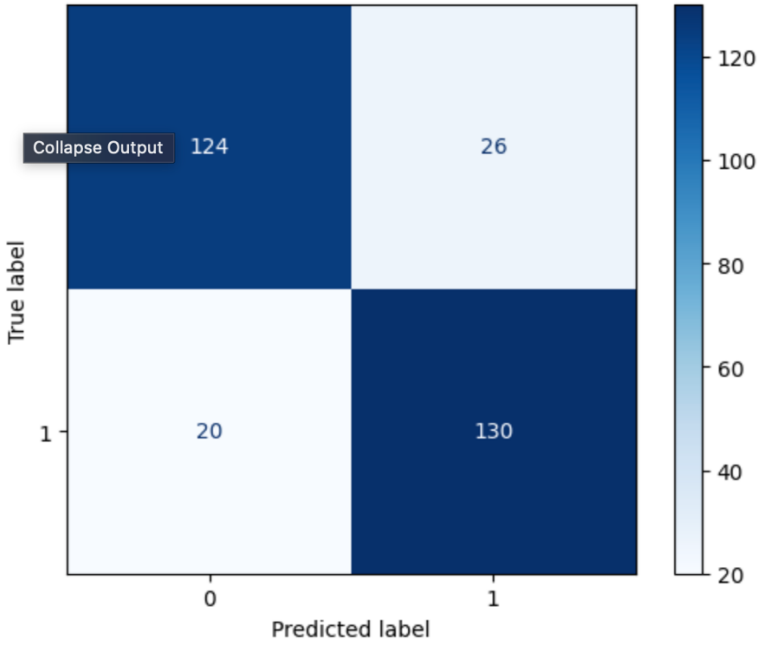
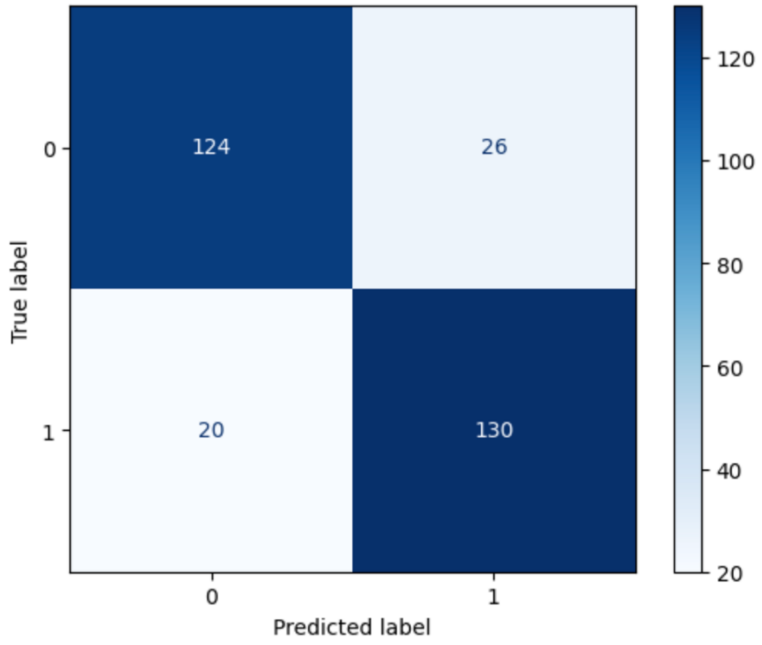


Figure 28

Confusion Matrix using SVM (linear) with cross validation and split 70/30



As can be seen the SVM showed better result when the data was spitted 70/30. In contrast, the other models Random Forest and XGB show a better performance using 80/20.

Additionally, the following hyper parameter tunning is the best for each model. This means shows the best performance.

Table 2

Hyper parameter tunning Random Forest, XGB and SVM(Linear)

	Parameters
Random Forest	criterion=entropy max_depth=none min_samples_split=10
XGB	comsample_bytree=0.8 learning_rate=0.3 max_depth=3 min_child_weight=1 sumsample = 0.6
SVM (linear),	Kernel=linear C=1 Gamma=1

Feature Importance

After having run the models, it was used Gini importance and Permutation importance to have a better intuition of which features are most important for a model.

Gini Importance: This method is used with trees models such as Random Forest or XGBoost and measures how much each feature is scored and helps to classify better.

Permutation Importance: This method is more reliable and robusta and measures the decrease in the model's score when the values of a single feature are randomly shuffled (permuted), showing how crucial that feature is to the model's predictions. This can be used with any model.

For this assignment, Random Forest and XGB are using both methods, while SVM is using only permutation importance.

Random Forest

Figure 29

Tables Gini and Permutation importance for Random Forest model

	Feature	Gini_Importance
16	contrast	0.139302
19	correlation	0.133023
12	kurtosis_cr_residual	0.107524
17	homogeneity	0.097070
13	kurtosis_cb_residual	0.092356
2	spectrum_flatness	0.087455
1	edge_density	0.077129
18	energy	0.030287
14	kurtosis_cb	0.026873
9	r_g_corr	0.021761
15	kurtosis_cr	0.021368
10	r_b_corr	0.020659
0	mean_intensity	0.020372
3	r_kurtosis	0.020008
4	g_kurtosis	0.018588
11	g_b_corr	0.018293
5	b_kurtosis	0.018201
6	r_skew	0.018160
7	g_skew	0.016410
8	b_skew	0.015163

	Feature	Permutation_Importance
12	kurtosis_cr_residual	0.062667
2	spectrum_flatness	0.051500
16	contrast	0.024500
17	homogeneity	0.023833
19	correlation	0.021667
13	kurtosis_cb_residual	0.014333
5	b_kurtosis	0.004333
4	g_kurtosis	0.001833
18	energy	0.000333
8	b_skew	0.000000
11	g_b_corr	-0.000167
3	r_kurtosis	-0.001333
7	g_skew	-0.001333
0	mean_intensity	-0.002333
6	r_skew	-0.002333
10	r_b_corr	-0.002333
15	kurtosis_cr	-0.002500
9	r_g_corr	-0.003833
14	kurtosis_cb	-0.004500
1	edge_density	-0.005000

Gini Importance (Top Table):

- The most important features for Gini are contrast (0.139), correlation (0.133), and kurtosis_cr_residual (0.108).
- The other feature that are no mentioned have a positive value. They indicate that are contributing to the model's performance.

Permutation Importance (Bottom Table):

- The most important features for Permutation are kurtosis_cr_residual (0.063), spectrum_flatness (0.051), and contrast (0.025).
- Negative values like edge_density and kurtosis_cb indicate that shuffling those features did not decrease the model's score and, in some cases, they could have slightly improved it, suggesting the model may not be using these features effectively or that they are mostly noise.

XGBoost

Figure 30

Tables Gini and Permutation importance for XGB model

	Feature	Gini_Importance
16	contrast	0.308114
19	correlation	0.130614
12	kurtosis_cr_residual	0.074034
2	spectrum_flatness	0.068355
17	homogeneity	0.056378
13	kurtosis_cb_residual	0.055006
1	edge_density	0.034361
5	b_kurtosis	0.033388
0	mean_intensity	0.030545
18	energy	0.030088
11	g_b_corr	0.026553
10	r_b_corr	0.025493
7	g_skew	0.023041
9	r_g_corr	0.022860
15	kurtosis_cr	0.015979
4	g_kurtosis	0.015285
14	kurtosis_cb	0.015059
8	b_skew	0.013945
6	r_skew	0.010650
3	r_kurtosis	0.010251

	Feature	Permutation_Importance
19	correlation	0.093667
12	kurtosis_cr_residual	0.082167
17	homogeneity	0.079333
2	spectrum_flatness	0.066000
16	contrast	0.024000
13	kurtosis_cb_residual	0.008000
5	b_kurtosis	0.004500
4	g_kurtosis	0.003500
11	g_b_corr	0.003000
10	r_b_corr	0.002500
1	edge_density	0.002000
6	r_skew	0.001833
14	kurtosis_cb	0.000333
0	mean_intensity	-0.000167
7	g_skew	-0.002167
9	r_g_corr	-0.002500
8	b_skew	-0.003167
3	r_kurtosis	-0.004167
18	energy	-0.004333
15	kurtosis_cr	-0.006333

Gini Importance (Top Table):

- The most important features for Gini are contrast (0.308), correlation (0.131), and kurtosis_cr_residual (0.074).
- Feature contrast is now significantly more dominant (over double the importance of the next feature) compared to the previous run.

Permutation Importance (Bottom Table):

- The most important features for Permutation are correlation (0.094), kurtosis_cr_residual (0.082), and homogeneity (0.079).
- While contrast was the clear winner in Gini table, it is in the 4th position in Permutation Importance (0.024). This means that the high Gini score may be due to it being selected early in the tree-building process rather than being the most predictively valuable feature overall.
- Negative values are again present for several features such as energy and kurtosis_cr. Indicating they may not be contributing positively to the model's performance on the validation data.

SVM (linear)

Figure 31

Tables Permutation importance for SVM model

	Feature	Permutation_Importance
16	contrast	0.103167
17	homogeneity	0.074167
12	kurtosis_cr_residual	0.067167
5	b_kurtosis	0.017500
8	b_skew	0.014333
13	kurtosis_cb_residual	0.012500
19	correlation	0.008833
11	g_b_corr	0.001333
4	g_kurtosis	0.001167
1	edge_density	0.001000
15	kurtosis_cr	0.000833
18	energy	0.000167
6	r_skew	0.000167
0	mean_intensity	0.000000
14	kurtosis_cb	-0.000500
3	r_kurtosis	-0.000500
2	spectrum_flatness	-0.000500
7	g_skew	-0.001000
10	r_b_corr	-0.003833
9	r_g_corr	-0.004167

- Contrast is in the top with an importance score of 0.103. This means that shuffling the contrast values causes better performance in the model. Thus, making it the most critical feature for the model's predictions.
- The features homogeneity (0.074) and kurtosis_cr_residual (0.067) show a high importance for the model.
- There is a significant drop in importance after the top three features. The 4th feature, b_kurtosis (0.0175), is less than a quarter as important as the 3rd feature, kurtosis_cr_residual.

- Features like mean_intensity (0) kurtosis_cb (-0.000500), and r_g_corr (0.004167) have little to no positive impact.
- The negative scores may indicate that randomly shuffling these features either had no effect or resulted in a slight improvement in the model's score. This suggests the model is not depending on these features for better predictions. Probably they are taken as noisy.

Conclusions

- This assignment allowed to learn how images are made by IA and how different they are from real images.
- The important forensics, which allow to manipulate these images to get the features correctly, this data at the end will be the most important resource of our models.
- Each model shows a high percentage of F1-score and accuracy and evaluation were required to avoid high bias or overfitting.
- It is important to mention that SVM was used first with BRF kernel, but this one showed a bad performance with the data collected. However, using linear kernel was enough to generalize the data and get a high score in metrics.
- Methods like Gini and Permutation importance allow to have a better understanding of how the features support the model prediction. This way, it is possible to tune the model, deleting some features that are not useful for some models.

References

GN, S. (2026, 4 06). *quarkml*. Retrieved from quarkml: https://www.quarkml.com/2022/10/the-rbf-kernel-in-svm-complete-guide.html

Kenton, W. (2025, 06 06). *investopedia*. Retrieved from investopedia: https://www.investopedia.com/terms/k/kurtosis.asp

Shafi, A. (2025, 21 31). *datacamp*. Retrieved from datacamp: https://www.datacamp.com/tutorial/random-forests-classifier-python

xgboosting. (n.d.). Retrieved from xgboosting: https://xgboosting.com/grid-search-xgboost-hyperparameters/