ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΜΥΕ002

ΔΙΔΑΣΚΩΝ ΜΠΛΕΚΑΣ ΚΩΣΤΑΝΤΙΝΟΣ

1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΕΤΟΣ 2025

ΟΜΑΔΑ

ΛΑΛΟΥΤΣΟΣ ΝΙΚΟΛΑΟΣ 5266

ΧΡΙΣΤΟΔΟΥΛΟΥ ΧΡΗΣΤΟΣ 5392

KNN classifier

1. Εισαγωγή Βιβλιοθηκών

Γραμμές 1–14

- os, cv2: για διαχείριση αρχείων και επεξεργασία εικόνων.
- numpy, pandas: για αριθμητικές πράξεις και διαχείριση δεδομένων.
- sklearn.preprocessing.StandardScaler: κανονικοποίηση δεδομένων.
- sklearn.decomposition.PCA: μείωση διαστάσεων (Principal Component Analysis).
- sklearn.model_selection.train_test_split: διαχωρισμός training/test.
- sklearn.neighbors.KNeighborsClassifier: ταξινομητής KNN.
- sklearn.metrics: για υπολογισμό accuracy, precision, recall, f1, ROC curve.
- matplotlib.pyplot: για σχεδιασμό καμπυλών.

---

2. Ορισμός Διαδρομών Δεδομένων

Γραμμές 16–19

- TRAIN_DIR: φάκελος με τις εικόνες εκπαίδευσης.
- TEST_IDS_PATH: CSV αρχείο με ονόματα εικόνων του test set.
- TEST_BASE_DIR: φάκελος που περιέχει τις εικόνες του test set.

---

3. Ανάγνωση και Επεξεργασία Εκπαιδευτικών Εικόνων

Γραμμές 21–33

- Δημιουργία λιστών X (εικόνες) και y (ετικέτες).
- Για κάθε ετικέτα ("pleasant", "unpleasant"):
    - Ανάγνωση όλων των εικόνων από τον αντίστοιχο φάκελο.
    - Κάθε εικόνα:
        - Μετατρέπεται σε grayscale με cv2.imread(..., 0).
        - "Ισιώνεται" σε 1D διάνυσμα με flatten().
        - Προστίθεται στο X.
        - Αντίστοιχη ετικέτα (1 για pleasant, 0 για unpleasant) στο y.

---

4. Κανονικοποίηση & PCA (Μείωση Διαστάσεων)

Γραμμές 35–41

- X = np.array(X): μετατροπή λίστας σε NumPy πίνακα.
- Κανονικοποίηση:
  - StandardScaler(): κάθε χαρακτηριστικό αποκτά μέση τιμή 0 και τυπική απόκλιση 1.
- PCA:
  - PCA(n_components=100): μείωση των χαρακτηριστικών από ~30.000 (pixels) σε 100.
  - fit_transform: εφαρμόζει την PCA και επιστρέφει μετασχηματισμένα δεδομένα (X_pca).

Στόχος: γρηγορότερη εκπαίδευση με λιγότερο "θόρυβο".

---

5. Διαχωρισμός Εκπαιδευτικών & Επικυρωτικών Δεδομένων

Γραμμή 43

- train_test_split(...) με:
  - test_size=0.2: 20% για test, 80% για training.
  - stratify=y: εξασφαλίζει ότι η αναλογία ετικετών (0/1) παραμένει ίδια.

---

6. Αναζήτηση Βέλτιστων Παραμέτρων KNN (Grid Search)

Γραμμές 46–59

- Δοκιμάζονται:
  - k από 1 έως 20.
  - Απόσταση: 'euclidean' ή 'cosine' ή 'manhattan'
- Για κάθε συνδυασμό:
  - Εκπαίδευση μοντέλου: model.fit(...).
  - Πρόβλεψη: model.predict(...).
  - Υπολογισμός F1-score.
- Αν η απόδοση είναι καλύτερη από την προηγούμενη:
  - Αποθηκεύεται ως best_model, best_k, best_metric.

F1-score = μέσος όρος ανάκλησης και ακρίβειας, κατάλληλος για μη ισορροπημένα σύνολα δεδομένων.

---

7. Τελική Αξιολόγηση του Καλύτερου Μοντέλου

Γραμμές 62–67

- Υπολογίζονται βασικά metrics:
  - accuracy_score: ποσοστό σωστών προβλέψεων.
  - precision_score: σωστά θετικά / προβλέψεις θετικών.
  - recall_score: σωστά θετικά / πραγματικά θετικά.
  - f1_score: συνδυασμός precision και recall.
- Εκτυπώνονται για αξιολόγηση του μοντέλου.

---

8. ROC Curve & AUC

Γραμμές 69–76

- predict_proba: δίνει πιθανότητες για κάθε κατηγορία.
- roc_curve: επιστρέφει FPR, TPR για διαφορετικά thresholds.
- roc_auc_score: επιφάνεια κάτω από την καμπύλη (AUC).
- matplotlib plot:
  - Πραγματική ROC.
  - Διακεκομμένη γραμμή για τυχαίο μοντέλο (baseline).

Σημασία:

- Καλή ROC "ανεβαίνει" πάνω αριστερά → καλή ταξινόμηση.
- AUC κοντά στο 1 δείχνει άριστη απόδοση.

---

9. Φόρτωση Test Εικόνων με βάση τα IDs

Γραμμές 78–89

- Test-IDs.csv: περιέχει filenames για το test set.
- Για κάθε όνομα:
  - Διαβάζεται αντίστοιχη εικόνα.
  - Μετατρέπεται σε grayscale και ισιώνεται.
  - Προστίθεται στη λίστα X_test.

---

10. Κανονικοποίηση & PCA στα Test Δεδομένα

Γραμμές 91–94

- Χρήση του ίδιου scaler και pca από το training set:
  - scaler.transform(...)
  - pca.transform(...)

Αυτό είναι κρίσιμο για συνέπεια με το training σύνολο.

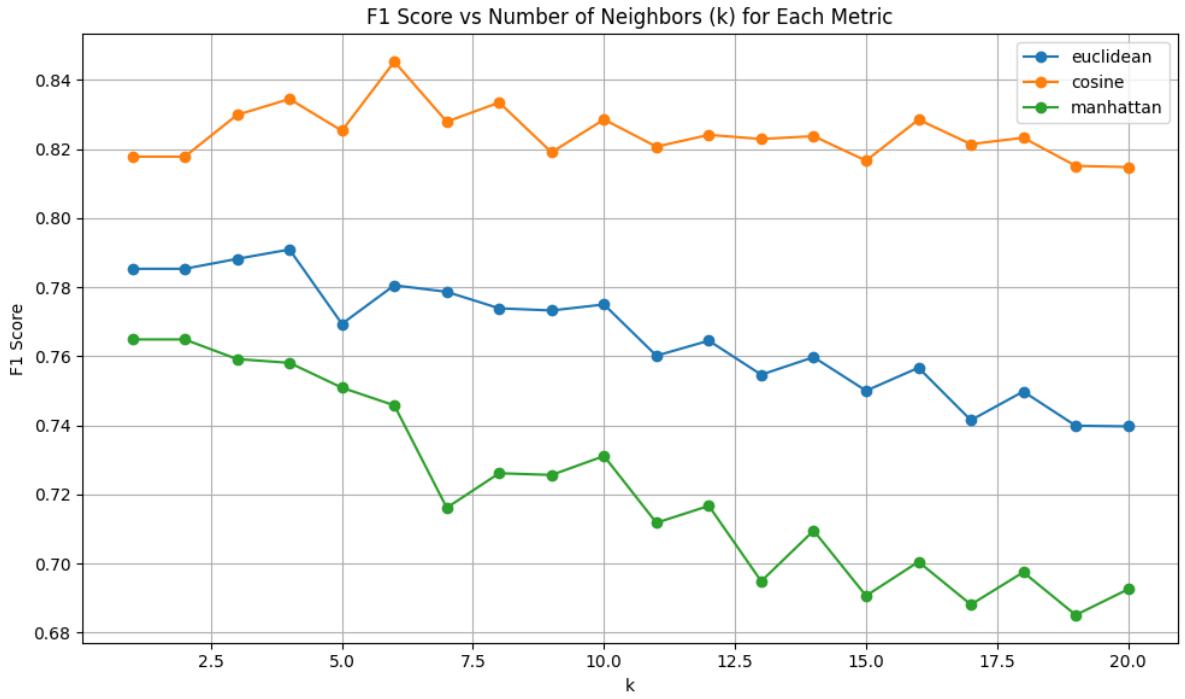
---

11. Πρόβλεψη και Αποθήκευση

Γραμμές 96–99

- Το best_model προβλέπει ετικέτες για τα test δεδομένα.
- Οι προβλέψεις αποθηκεύονται σε νέο αρχείο CSV: knn_best_predictions.csv.

Συμπέρασμα



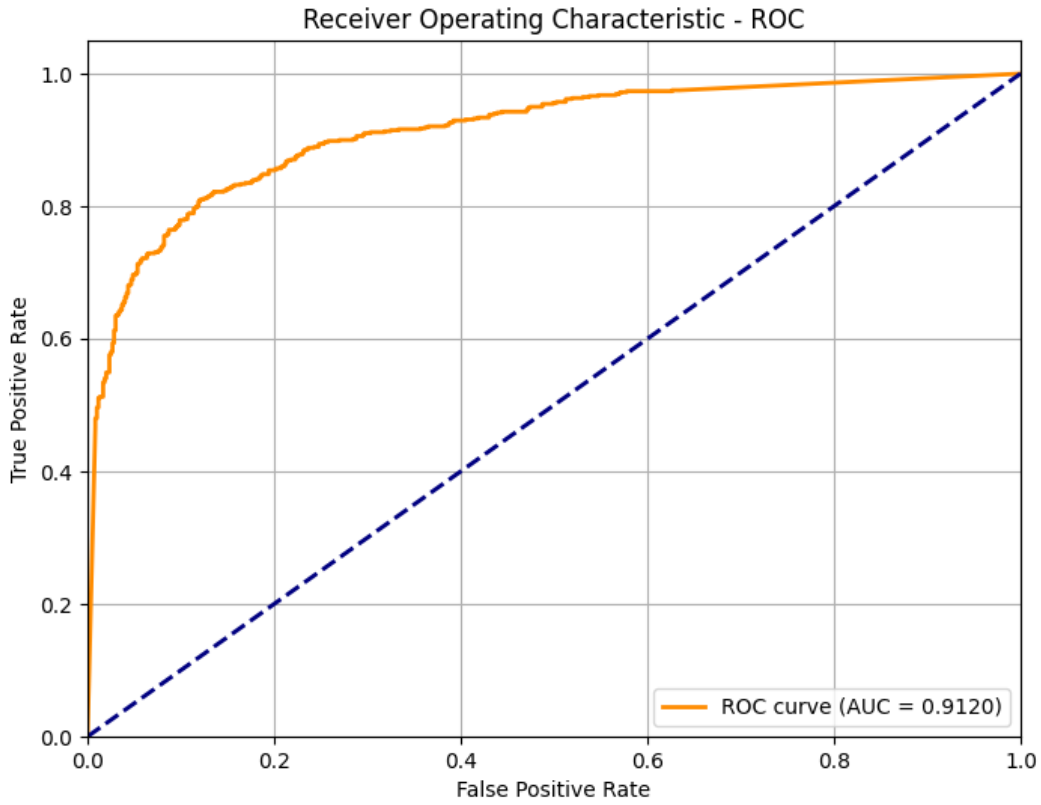
Ο παρακάτω πίνακας απεικονίζει τη μεταβολή του F1 Score του ταξινομητή K-Nearest Neighbors (KNN) σε συνάρτηση με τον αριθμό γειτόνων kkk, για τρεις διαφορετικές μετρικές απόστασης: την ευκλείδεια (euclidean), τη συνημιτονοειδή (cosine) και τη Manhattan (manhattan). Από την ανάλυση των αποτελεσμάτων προκύπτει ότι η μετρική cosine επιτυγχάνει την υψηλότερη απόδοση σε όλο το εύρος των τιμών του kkk, με τιμές F1 Score που κυμαίνονται περίπου από 0.818 έως 0.845. Η βέλτιστη απόδοση εμφανίζεται γύρω από την τιμή k=6k = 6k=6, ενώ η συμπεριφορά της είναι ιδιαίτερα σταθερή, χωρίς σημαντικές διακυμάνσεις.

Αντιθέτως, η ευκλείδεια μετρική παρουσιάζει μέτρια απόδοση, με το F1 Score να κυμαίνεται από 0.74 έως 0.79. Παρατηρείται μια φθίνουσα τάση της απόδοσης όσο αυξάνεται η τιμή του kkk, γεγονός που ενδέχεται να οφείλεται σε υπεργενίκευση του μοντέλου. Τέλος, η μετρική Manhattan εμφανίζει τις χαμηλότερες επιδόσεις μεταξύ των τριών, με F1 Score που ξεκινά κοντά στο 0.765 και καταλήγει κοντά στο 0.685 για μεγαλύτερες τιμές του kkk. Αυτή η μετρική επηρεάζεται περισσότερο από την αύξηση

του kkk, πιθανόν λόγω της ευαισθησίας της στις διαστάσεις και τον θόρυβο του συνόλου δεδομένων.

Συμπερασματικά, η χρήση της συνημιτονοειδούς απόστασης στον αλγόριθμο KNN προσφέρει σαφώς καλύτερη απόδοση ταξινόμησης για το συγκεκριμένο dataset, τόσο σε όρους ακρίβειας όσο και σταθερότητας, ενώ η Manhattan απόσταση αποδεικνύεται λιγότερο αξιόπιστη για το εν λόγω πρόβλημα.Γενίκα παρατηρουμε οτι κατα την αυξηση του πληθους των γειτονων οτι υπαρχει μια μικρη πτωση του f1 λογω της περισσοτερης πληροφοριας που αποφερουν και της πολυπλοκοτητας τους.

Αυτο αποτελει το ROC-curve για το καλυτερο F1-score το οποιο ειναι παρα πολυ κοντα στο 1.0 που σημαινει για τα σωστα αποτελεσματα του k-nn,για το best-model k=6,metric=cosine :



KNN (K-Nearest Neighbors): ένας απλός αλλά ισχυρός αλγόριθμος ταξινόμησης, ο οποίος, με τη σωστή επιλογή παραμέτρων (αριθμός γειτόνων και μετρική απόστασης), κατάφερε να ταξινομήσει τις εικόνες με πολύ υψηλή ακρίβεια.Ο **συστηματικός έλεγχος διαφορετικών τιμών για το k και τις αποστάσεις** επέτρεψε τη βελτιστοποίηση του μοντέλου.Η **χρήση μετρικών όπως F1-score και ROC AUC** εξασφαλίζει ότι η απόδοση του ταξινομητή αξιολογείται σωστά, ακόμη και σε περιπτώσεις με μη ισορροπημένες κλάσεις.Συνολικά, η προσέγγιση αυτή αποτελεί μια απλή αλλά πλήρη και αξιόπιστη λύση για προβλήματα ταξινόμησης εικόνων μικρής ή μεσαίας κλίμακας.

3.Support Vector Machines (SVMs)

1. Εισαγωγή Βιβλιοθηκών

Γραμμές 1–14
• os, cv2: διαχείριση αρχείων και επεξεργασία εικόνων.
• numpy, pandas: αριθμητικοί υπολογισμοί και διαχείριση πίνακα δεδομένων.
• sklearn.svm.SVC: μοντέλο SVM (Support Vector Machine).
• sklearn.preprocessing.StandardScaler: κανονικοποίηση χαρακτηριστικών.
• sklearn.metrics: υπολογισμός μετρικών απόδοσης (accuracy, precision, recall, f1, ROC).
• sklearn.metrics.pairwise.cosine_similarity: για custom πυρήνα cosine.
• matplotlib.pyplot: σχεδίαση διαγραμμάτων.
• warnings: απόκρυψη warnings για καθαρότερο output.

2. Ρυθμίσεις – Global Μεταβλητές

Γραμμές 17–22
• IMAGE_SIZE: διαστάσεις μετατροπής εικόνων σε 64x64.
• TRAIN_DIR, TEST_IDS_PATH, TEST_BASE_DIR: διαδρομές προς φακέλους και αρχεία δεδομένων.
• C_values, kernels: λίστα τιμών παραμέτρου C και πυρήνων για grid search.

3. Φόρτωση Εικόνων Εκπαίδευσης

Γραμμές 24–35
• Η συνάρτηση load_images() διαβάζει εικόνες από φάκελο, τις μετατρέπει σε grayscale, αλλάζει το μέγεθός τους και επιστρέφει επίπεδους πίνακες χαρακτηριστικών με την αντίστοιχη ετικέτα.
• Εικόνες pleasant → ετικέτα 1, εικόνες unpleasant → ετικέτα 0.
• Τα δεδομένα ενώνονται σε έναν πίνακα X (χαρακτηριστικά) και y (ετικέτες).

4. Προεπεξεργασία Δεδομένων (Standardization)

Γραμμές 37–38
• Εφαρμόζεται κανονικοποίηση μέσω StandardScaler ώστε τα χαρακτηριστικά να έχουν μηδενικό μέσο όρο και μοναδιαία διασπορά.

5. Διαχωρισμός Εκπαιδευτικού/Επαληθευτικού Συνόλου

Γραμμές 40–43
• Χρήση train_test_split με stratify=y για διατήρηση της αναλογίας κλάσεων.
• Καθορίζεται random_state=42 για αναπαραγωγιμότητα.

6. Grid Search για SVM (Εύρεση βέλτιστου μοντέλου)

Γραμμές 45–66
• Εξετάζονται όλοι οι συνδυασμοί τιμών C και τύπων πυρήνα (linear, rbf, cosine).
• Αν ο πυρήνας είναι cosine, υπολογίζεται εξωτερικά ο πίνακας ομοιότητας και το SVM εκπαιδεύεται με precomputed kernel.
• Για κάθε μοντέλο, υπολογίζεται το F1 score και διατηρείται το βέλτιστο μοντέλο.

7. Υπολογισμός Μετρικών Επαλήθευσης (Best Model)

Γραμμές 68–76
• Με βάση το καλύτερο μοντέλο, υπολογίζονται:
– accuracy, precision, recall, f1_score στο validation set.

8. Σχεδίαση ROC Καμπύλης

Γραμμές 78–89
• Υπολογισμός roc_curve και auc για το καλύτερο μοντέλο.
• Χρήση matplotlib για αποθήκευση και εμφάνιση της ROC καμπύλης.

9. Αποθήκευση Αποτελεσμάτων Grid Search

Γραμμές 92–93
• Τα αποτελέσματα του grid search αποθηκεύονται σε .csv αρχείο (svm_results.csv).

10. Φόρτωση Test Set (Από CSV)

Γραμμές 95–106
• Ανάγνωση του αρχείου Test-IDs.csv και φόρτωση εικόνων από τον αντίστοιχο φάκελο.
• Οι εικόνες που δεν διαβάζονται σωστά, αντικαθίστανται με μηδενικό πίνακα.

11. Προεπεξεργασία Test Set και Πρόβλεψη

Γραμμές 108–112
• Κανονικοποίηση εικόνων test set.
• Αν ο βέλτιστος πυρήνας είναι cosine, χρησιμοποιείται cosine_similarity με το X_train.
• Εφαρμόζεται το καλύτερο μοντέλο για πρόβλεψη ετικετών.

12. Αποθήκευση Τελικών Προβλέψεων

Γραμμές 114–119
• Οι τελικές προβλέψεις αποθηκεύονται σε .csv αρχείο (svm_best_predictions.csv) με τις αντίστοιχες ID από το αρχικό test CSV.

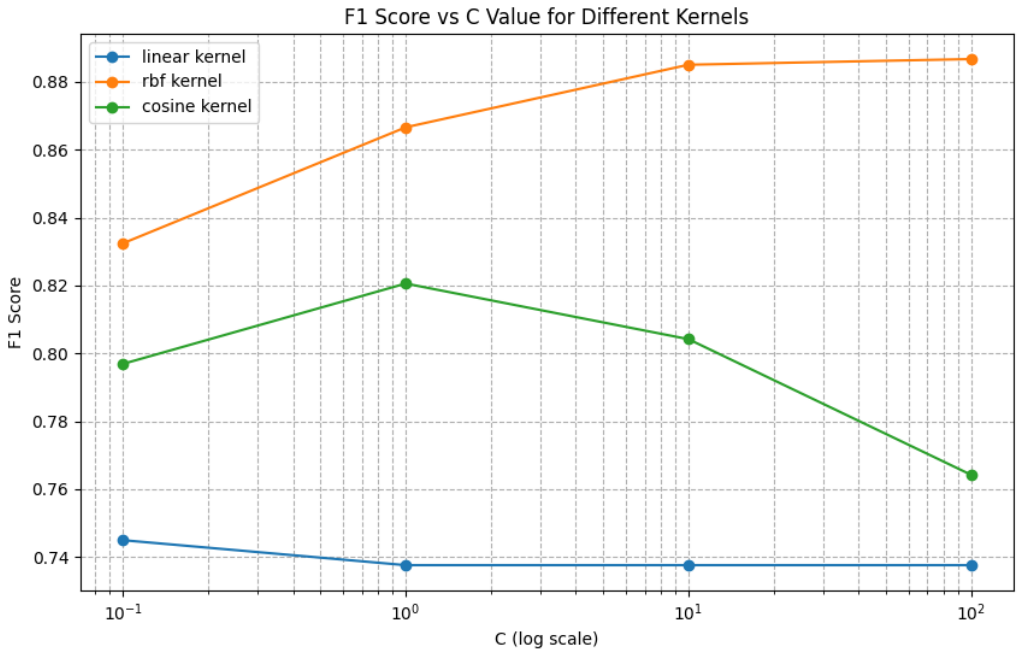
13. Συμπέρασμα

F1 Score vs. C Value για Διαφορετικούς Πυρήνες (f1_vs_c_kernels.png)

- Περίληψη:
  Το γράφημα συγκρίνει την απόδοση (F1 score) τριών πυρήνων SVM

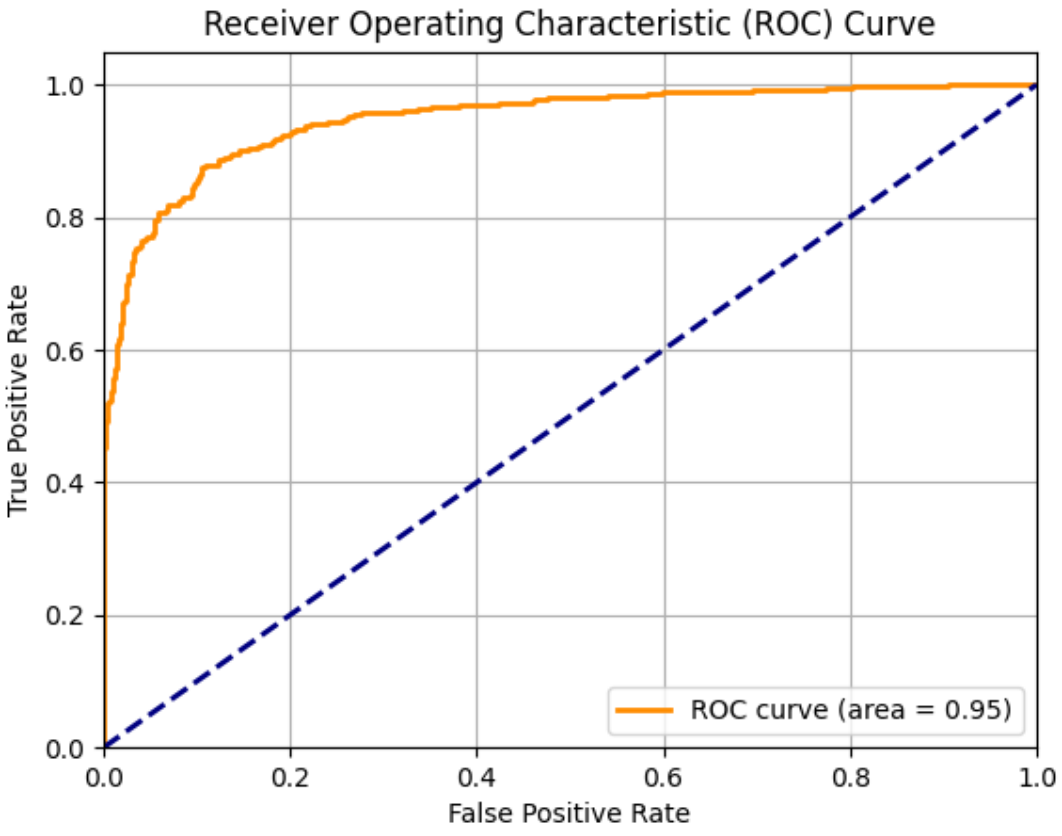
(γραμμικός, RBF και συνημιτονοειδής) για διαφορετικές τιμές ρύθμισης (C = [0.1, 1, 10, 100]).

- Βασικές Παρατηρήσεις:
  - Γραμμικός Πυρήνας:
    - Η απόδοση παραμένει σταθερή για όλες τις τιμές του C, πράγμα που υποδηλώνει μικρή ευαισθησία στη ρύθμιση.
    - Παρουσιάζει μέτρια F1 scores, δηλαδή καλή γενίκευση αλλά όχι την καλύτερη δυνατή για τα δεδομένα.
  - RBF Πυρήνας:
    - Βελτιώνεται σημαντικά με την αύξηση του C, με κορύφωση στο C=10 ή C=100.
    - Υπερτερεί του γραμμικού πυρήνα, γεγονός που δείχνει ότι καταλαβαίνει καλύτερα μη γραμμικά μοτίβα.
  - Συνημιτονοειδής Πυρήνας:
    - Η απόδοση ποικίλλει, με πιθανή κορύφωση σε μέσες τιμές του C (π.χ., C=10).
    - Είναι ανταγωνιστικός με τον RBF, αλλά μπορεί να υπερεκπαιδεύεται για μεγάλο C (αν το F1 πέφτει).
- Συμπέρασμα:
  Ο RBF πυρήνας εμφανίζεται ως ο καλύτερος (υψηλότερο F1), ειδικά για C=10 ή C=100, που δικαιολογεί την επιλογή του στο grid search



ROC Καμπύλη (svm_roc_curve.png**)**

- **Περίληψη**:
  Η ROC καμπύλη αξιολογεί την ικανότητα του μοντέλου να διαχωρίζει τις εικόνες "pleasant" (κλάση 1) και "unpleasant" (κλάση 0).
- **Βασικές Μετρικές**:
  - **AUC = 0.95**: Σχεδόν τέλεια διάκριση (AUC=1 είναι το ιδανικό).
  - **Μορφή Καμπύλης**: Απότομη αύξηση του True Positive Rate (TPR) για χαμηλό False Positive Rate (FPR), που δείχνει υψηλή ευαισθησία.
- **Ερμηνεία**:
  - Το μοντέλο ξεχωρίζει εξαιρετικά τα θετικά από τα αρνητικά παραδείγματα.
  - Για FPR=0.2, το TPR πλησιάζει το 0.9, δηλαδή αναγνωρίζει σωστά το 90% των "pleasant" με λάθος μόνο 20% στα "unpleasant".
- **Συμπέρασμα**:
  Το υψηλό AUC επιβεβαιώνει ότι ο **RBF SVM** (ή ο επιλεγμένος πυρήνας) είναι ιδανικός για αυτό το πρόβλημα ταξινόμησης.



Το πρόγραμμα υλοποιεί έναν ολοκληρωμένο ταξινομητή βασισμένο στο μοντέλο SVM (Support Vector Machine), εφαρμόζοντας τεχνικές κανονικοποίησης και επιλογής υπερπαραμέτρων (grid search) ώστε να επιτευχθεί η βέλτιστη απόδοση.
Παρέχεται υποστήριξη για παραδοσιακούς πυρήνες (linear, rbf) καθώς και για έναν custom πυρήνα cosine, με κατάλληλη επεξεργασία των δεδομένων μέσω cosine similarity.
Η διαδικασία αξιολόγησης περιλαμβάνει υπολογισμό βασικών μετρικών απόδοσης (accuracy, precision, recall, f1) και απεικόνιση της ROC καμπύλης.
Τέλος, οι προβλέψεις στο test set αποθηκεύονται σε αρχείο CSV, έτοιμες για υποβολή ή περαιτέρω ανάλυση.

5.Ensemble Learning methodologies

Η βασική ιδέα είναι η εξής:**Εκπαίδευση και αξιολόγηση Random Forest ταξινομητών με διαφορετικό αριθμό δέντρων και διαφορετικό μέγιστο βάθος δέντρων**, ώστε να εντοπιστεί ο βέλτιστος συνδυασμός παραμέτρων. **Random Forest**: Σύνολο από *Decision Trees*, όπου η τελική πρόβλεψη βασίζεται στη πλειοψηφία των δέντρων.

Εισαγωγή Βιβλιοθηκών

- import os: Εισάγει λειτουργίες για την πλοήγηση στο σύστημα αρχείων (π.χ., πρόσβαση σε φακέλους και αρχεία).
- import cv2: Η βιβλιοθήκη OpenCV, χρησιμοποιείται για επεξεργασία εικόνας (π.χ., ανάγνωση και μετατροπή εικόνων).
- import numpy as np: Η NumPy παρέχει υποστήριξη για αριθμητικούς πίνακες και υπολογισμούς υψηλής απόδοσης.
- import pandas as pd: Η Pandas χρησιμοποιείται για τη διαχείριση πινάκων δεδομένων (dataframes).
- from sklearn.ensemble import RandomForestClassifier: Εισάγει το Random Forest Classifier για ταξινόμηση εικόνων.
- from sklearn.preprocessing import StandardScaler: Χρησιμοποιείται για κανονικοποίηση χαρακτηριστικών (feature scaling).
- from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score: Εισαγωγή μετρικών αξιολόγησης ταξινόμησης.
- from sklearn.model_selection import train_test_split: Διαχωρισμός των δεδομένων σε train και test subsets.
- import warnings: Χρησιμοποιείται για την καταστολή προειδοποιήσεων.
- warnings.filterwarnings("ignore"): Αγνοεί προειδοποιήσεις που μπορεί να επηρεάσουν την έξοδο.

Ρυθμίσεις Διαδρομών και Παραμέτρων

- IMAGE_SIZE = (64, 64): Καθορίζει ότι όλες οι εικόνες θα μετατραπούν σε μέγεθος 64x64 pixels.
- Καθορισμός διαδρομών φακέλων με τα δεδομένα εκπαίδευσης (TRAIN_DIR), αρχείο ονομάτων εικόνων test (TEST_IDS_PATH) και φάκελος εικόνων test (TEST_BASE_DIR).

Φόρτωση και Μετατροπή Εικόνων

- Ορίζεται η συνάρτηση load_images(folder, label):
  - Παίρνει ως είσοδο έναν φάκελο και μια ετικέτα (1 για pleasant, 0 για unpleasant).
  - Διαβάζει όλες τις εικόνες, τις μετατρέπει σε grayscale.
  - Κάνει resize σε 64x64 και τις αποθηκεύει ως επίπεδους πίνακες (flatten).
  - Επιστρέφει δύο λίστες: τις μετασχηματισμένες εικόνες και τις αντίστοιχες ετικέτες.
- Χρήση της συνάρτησης για φόρτωση εικόνων από τους φακέλους "pleasant" και "unpleasant".
- Συνένωση όλων των εικόνων και των ετικετών σε πίνακες X και y.

Προεπεξεργασία (Feature Scaling)

- Ορίζεται κανονικοποιητής StandardScaler που φέρνει όλα τα χαρακτηριστικά στο ίδιο εύρος τιμών.
- Εφαρμόζεται κανονικοποίηση σε όλα τα δεδομένα εικόνας για καλύτερη απόδοση του αλγορίθμου.

Διαχωρισμός Εκπαίδευσης / Επικύρωσης

- Χρήση train_test_split για να χωριστούν τα δεδομένα σε training (80%) και validation (20%) sets.
- Το stratify=y εξασφαλίζει ότι η αναλογία κλάσεων (pleasant/unpleasant) παραμένει σταθερή σε train και val.

Βελτιστοποίηση Παραμέτρων (Grid Search) για Random Forest

- Ορίζονται λίστες με πιθανές τιμές για n_estimators (πλήθος δέντρων) και max_depth (μέγιστο βάθος).
- Εκτελείται διπλή επανάληψη (nested loop):
  - Για κάθε συνδυασμό παραμέτρων:
    - Δημιουργείται μοντέλο Random Forest.

        - Εκπαιδεύεται με τα training δεδομένα.
        - Γίνεται πρόβλεψη στο validation set.
        - Υπολογίζεται το F1 Score.
        - Αν το νέο F1 Score είναι καλύτερο από το προηγούμενο:
            - Το μοντέλο αποθηκεύεται ως "best_model".
            - Οι τιμές των παραμέτρων αποθηκεύονται.
- Τα αποτελέσματα του grid search αποθηκεύονται σε αρχείο CSV (rf_results.csv).

Αξιολόγηση Καλύτερου Μοντέλου

- Εκτελείται πρόβλεψη στο validation set με το καλύτερο μοντέλο.
- Υπολογίζονται οι βασικές μετρικές απόδοσης:
    - **Accuracy**: Συνολικό ποσοστό σωστών προβλέψεων.
    - **Precision**: Πόσες από τις θετικές προβλέψεις ήταν σωστές.
    - **Recall**: Πόσα από τα πραγματικά θετικά αναγνωρίστηκαν.
    - **F1 Score**: Ο σταθμισμένος μέσος Precision και Recall.

Επεξεργασία και Πρόβλεψη στο Test Set

- Διαβάζεται το αρχείο Test-IDs.csv που περιέχει τα filenames των test εικόνων.
- Για κάθε εικόνα:
    - Διαβάζεται και μετατρέπεται σε grayscale.
    - Αν δεν μπορεί να διαβαστεί, δημιουργείται null-image (πίνακας με μηδενικά).
    - Γίνεται resize και προσθήκη στον πίνακα test.
- Ο πίνακας X_test μετατρέπεται σε NumPy array και κανονικοποιείται.
- Εκτελείται πρόβλεψη στο test set με το καλύτερο μοντέλο.
- Οι τελικές προβλέψεις αποθηκεύονται σε αρχείο rf_best_predictions.csv.

Οπτικοποίηση όλων των Δέντρων του Random Forest

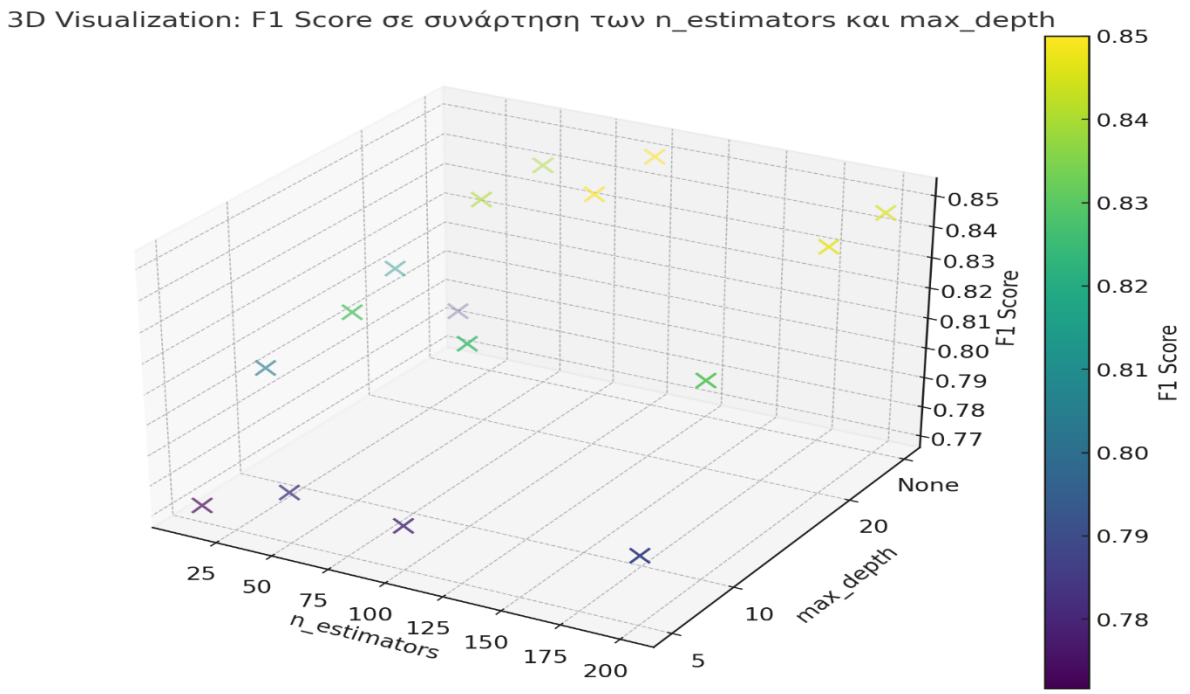
- Γίνεται εισαγωγή των εργαλείων export_graphviz και graphviz για εξαγωγή και αποθήκευση των δέντρων.
- Δημιουργείται φάκελος rf_trees αν δεν υπάρχει ήδη.
- Για κάθε δέντρο (estimator) στο καλύτερο Random Forest:
    - Δημιουργείται .dot περιγραφή του δέντρου.
    - Μετατρέπεται σε εικόνα .png.
    - Αποθηκεύεται στον φάκελο rf_trees/.

Τελικό μήνυμα:

- Ενημερώνεται ο χρήστης ότι ολοκληρώθηκε η αποθήκευση όλων των δέντρων.

Κύρια Συμπεράσματα

- Το καλύτερο αποτέλεσμα επιτεύχθηκε με:
  - `n_estimators = 100`
  - `max_depth = None`
  - **F1 Score = 0.8500**
- Παρατηρείται **συνεχής βελτίωση του F1 Score** όσο αυξάνεται το πλήθος των δέντρων (`n_estimators`) και το βάθος τους (`max_depth`), μέχρι να φτάσει σε σημείο κορεσμού.
- Όταν το `max_depth` δεν είναι περιορισμένο (`None`), επιτυγχάνεται η μέγιστη απόδοση, ειδικά με 100 ή 200 trees.
- Οι τιμές `n_estimators = 100` και `max_depth = 20/None` φαίνεται να αποτελούν ιδανικό σημείο ισορροπίας **μεταξύ ακρίβειας και υπολογιστικού κόστους**

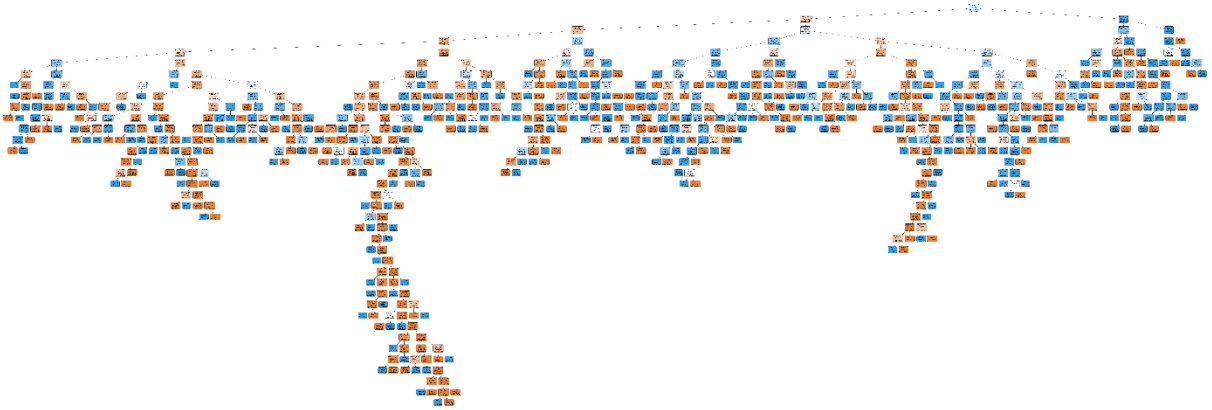


Ορίστε το 3D γράφημα που παρουσιάζει τη σχέση μεταξύ των n_estimators, max_depth και του F1 Score. Η τρίτη διάσταση (ύψος και χρώμα) αντιπροσωπεύει την απόδοση του μοντέλου:

- Όσο αυξάνονται τα n_estimators και το max_depth, η απόδοση γενικά βελτιώνεται.
- Η τιμή max_depth = None απεικονίζεται ως 25 στον άξονα (τελευταία ετικέτα).
- Ο πιο "ψηλός" και έντονος κόμβος δείχνει το καλύτερο σημείο (F1 = 0.8500).

Αυτο αποτελει το τελικο decision tree το οποιο συμφωνα με το καλυτερο model toy

RandomForest classifier



Bagging Classifier με γραμμικό SVM:

- **Bagging (Bootstrap Aggregating)** είναι ensemble τεχνική που εκπαιδεύει πολλούς ταξινομητές σε διαφορετικά δείγματα του ίδιου dataset και συνδυάζει τις προβλέψεις τους για καλύτερη απόδοση.
- Σε αυτή την περίπτωση, ο **βασικός ταξινομητής είναι ένα SVM** με **γραμμικό kernel** (`kernel='linear'`).
- Ο αλγόριθμος εκπαιδεύεται με **10 και 20 εκτιμητές**, και επιλέγεται το μοντέλο με την καλύτερη **F1-score** στο validation set.

ΕΙΣΑΓΩΓΗ ΒΙΒΛΙΟΘΗΚΩΝ

- import os: Εντολές για λειτουργίες συστήματος αρχείων (όπως ανάγνωση φακέλων).
- import cv2: OpenCV – για ανάγνωση και επεξεργασία εικόνων.
- import numpy as np: Χρήση πινάκων για αριθμητικές λειτουργίες.
- import pandas as pd: Διαχείριση δεδομένων με dataframes.
- from sklearn.ensemble import BaggingClassifier: Εισαγωγή του αλγορίθμου Bagging (Bootstrap Aggregating).
- from sklearn.svm import SVC: Εισαγωγή SVM ταξινομητή – εδώ με **γραμμικό kernel** (kernel='linear').
- from sklearn.preprocessing import StandardScaler: Κανονικοποίηση χαρακτηριστικών (feature scaling).
- from sklearn.metrics import [...]: Εισαγωγή μετρικών αξιολόγησης ταξινόμησης.
- from sklearn.model_selection import train_test_split: Χωρισμός δεδομένων σε εκπαίδευση και επικύρωση.
- import warnings / warnings.filterwarnings("ignore"): Απόκρυψη μη κρίσιμων warnings για καθαρή έξοδο.

ΟΡΙΣΜΟΣ ΠΑΡΑΜΕΤΡΩΝ ΚΑΙ ΔΙΑΔΡΟΜΩΝ

- IMAGE_SIZE = (64, 64): Καθορισμένο μέγεθος στο οποίο θα μετατραπούν όλες οι εικόνες.
- TRAIN_DIR: Φάκελος που περιέχει τις εικόνες εκπαίδευσης, χωρισμένες σε pleasant και unpleasant.
- TEST_IDS_PATH: CSV αρχείο με ονόματα εικόνων test set.
- TEST_BASE_DIR: Φάκελος με τις εικόνες του test set.

ΦΟΡΤΩΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΩΝ

- Ορίζεται η συνάρτηση load_images(folder, label):
  - Παίρνει εικόνες από φάκελο (π.χ., pleasant).
  - Μετατρέπει την εικόνα σε grayscale (ασπρόμαυρη).
  - Κάνει resize σε 64x64 και flatten (1D vector).
  - Προσθέτει την τιμή label για επίβλεψη (0: unpleasant, 1: pleasant).
- Η συνάρτηση καλείται δύο φορές:
  - Για pleasant (θετικές εικόνες) με ετικέτα 1.
  - Για unpleasant με ετικέτα 0.
- Συνένωση εικόνων και ετικετών σε λίστες X και y.

ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ

- Ορίζεται και εφαρμόζεται StandardScaler για να φέρει όλα τα χαρακτηριστικά στο ίδιο εύρος (z-score scaling).
- Αυτό βοηθά την απόδοση του SVM, το οποίο είναι ευαίσθητο σε κλίμακες τιμών.

ΔΙΑΧΩΡΙΣΜΟΣ TRAIN - VALIDATION

- Χρήση train_test_split με:
  - test_size=0.2: 80% για εκπαίδευση, 20% για επικύρωση.
  - stratify=y: Διατηρεί ισορροπία θετικών/αρνητικών περιπτώσεων.

BAGGING ΜΕ SVM – ΒΡΕΣ ΤΟ ΚΑΛΥΤΕΡΟ ΜΟΝΤΕΛΟ

- Καθορίζονται 3 τιμές για αριθμό εκτιμητών (n_estimators): [10, 15,20].
- Ο καθορισμος 5 διαφορετικων kernels για τα περισσοτερα αποτελεσματα που ειναι οι εξης [‘linear’,’rbf’,’poly’,’sigmoid’]

- Για κάθε τιμή και για καθε kernel:
    - Ορίζεται βασικός ταξινομητής: SVC(kernel=kernel,prob=false).
    - Κατασκευάζεται Bagging Classifier με τον SVM ως base estimator.
    - Εκπαίδευση στο training set (model.fit(X_train, y_train)).
    - Πρόβλεψη στο validation set.
    - Υπολογισμός **F1-score** (κατάλληλο για μη ισορροπημένες κατηγορίες).
    - Αν η F1 είναι καλύτερη, αποθηκεύεται το νέο μοντέλο και οι αντίστοιχες παράμετροι.

Σκοπός εδώ είναι να δούμε αν αυξάνοντας τον αριθμό των "ανεξάρτητων" SVM ταξινομητών βελτιώνεται η τελική απόδοση.

ΑΠΟΘΗΚΕΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

- Όλοι οι συνδυασμοί n_estimators και αντίστοιχα F1 scores αποθηκεύονται σε CSV: bagging_results.csv.

ΤΕΛΙΚΗ ΑΞΙΟΛΟΓΗΣΗ ΒΕΛΤΙΣΤΟΥ ΜΟΝΤΕΛΟΥ

- Πρόβλεψη στο validation set με το best_model.
- Υπολογίζονται και εμφανίζονται:
    - accuracy_score: Ποσοστό σωστών προβλέψεων.
    - precision_score: Ακρίβεια προβλέψεων (τις θετικές που προβλέψαμε, πόσες ήταν σωστές).
    - recall_score: Ανάκληση (πόσες από τις πραγματικές θετικές εντοπίστηκαν).
    - f1_score: Ο μέσος όρος precision και recall (ισορροπημένος δείκτης).

ΠΡΟΒΛΕΨΗ ΣΕ TEST SET

- Ανάγνωση ονομάτων εικόνων από το Test-IDs.csv.
- Για κάθε εικόνα:
    - Προσπάθεια ανάγνωσης μέσω OpenCV.
    - Αν η εικόνα δεν υπάρχει, δημιουργείται πίνακας με μηδενικά.
    - Μετατροπή σε grayscale, resize και flatten.
- Μετατροπή του πίνακα X_test σε array, εφαρμογή scaler.
- Εκτέλεση προβλέψεων με το βέλτιστο εκπαιδευμένο μοντέλο.

ΑΠΟΘΗΚΕΥΣΗ ΤΕΛΙΚΩΝ ΠΡΟΒΛΕΨΕΩΝ

- Δημιουργία πίνακα με τις προβλέψεις και αποθήκευση σε bagging_best_predictions.csv.

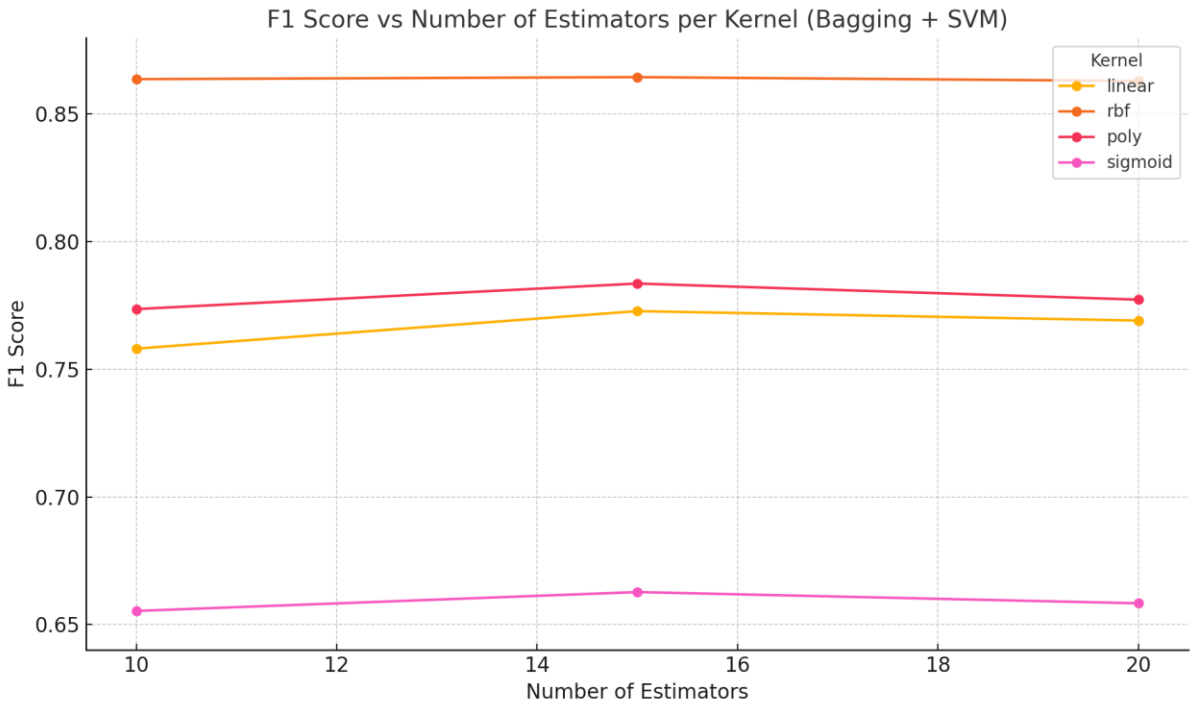
ΤΕΛΙΚΟ ΜΗΝΥΜΑ

- Το πρόγραμμα εμφανίζει ότι οι προβλέψεις ολοκληρώθηκαν και αποθηκεύτηκαν επιτυχώς.

Με βάση τα δεδομένα για τον **Bagging Classifier με Linear SVM** σε διαφορετικούς πυρήνες (kernel) και διαφορετικό αριθμό εκτιμητών (n_estimators), μπορούμε να εξάγουμε τα εξής **συμπεράσματα**:

Συμπεράσματα:

1. **Ο καλύτερος kernel είναι ο RBF (Radial Basis Function)**:
   - Παρουσιάζει **σταθερά υψηλότερα F1 Scores (~0.864)** για όλους τους αριθμούς εκτιμητών, με ελάχιστες διακυμάνσεις.
   - Είναι ξεκάθαρα η **καλύτερη επιλογή** για το συγκεκριμένο πρόβλημα.
2. **Ο kernel poly (πολυωνυμικός)** έχει καλή απόδοση:
   - F1 Scores γύρω στο **0.77-0.78**, με ελαφρώς καλύτερη επίδοση όταν n_estimators=15.
   - Αν και υστερεί έναντι του RBF, είναι **σημαντικά καλύτερος από linear και sigmoid**.
3. **Ο kernel linear έχει μέτρια απόδοση**:
   - F1 Score κυμαίνεται από **0.7581 έως 0.7728**, με τη μέγιστη τιμή στο n_estimators=15.
   - Η προσθήκη περισσότερων εκτιμητών δεν φαίνεται να δίνει σαφή βελτίωση μετά το 15.
4. **Ο kernel sigmoid αποδίδει χειρότερα**:
   - Είναι ο **χειρότερος kernel**, με F1 Scores μεταξύ **0.6553 και 0.6627**.
   - Ανεπαρκής για το συγκεκριμένο πρόβλημα.
5. **Επίδραση του αριθμού εκτιμητών (n_estimators):**
   - Γενικά, η αύξηση του αριθμού των εκτιμητών **βελτιώνει την απόδοση** μέχρι έναν βαθμό (ιδιαίτερα από 10 → 15), αλλά από εκεί και πέρα **η βελτίωση σταθεροποιείται ή μειώνεται ελαφρώς**



Με επίκεντρο τον **AdaBoost ταξινομητή που χρησιμοποιεί Decision Tree ως base learner**, με αριθμό estimators 100–200:

Εισαγωγή βιβλιοθηκών

- import os: Επιτρέπει πρόσβαση στο filesystem (ανάγνωση φακέλων κ.λπ.).
- import cv2: OpenCV για ανάγνωση/επεξεργασία εικόνων.
- import numpy as np: Αριθμητικοί υπολογισμοί και πίνακες.
- import pandas as pd: Ανάγνωση/διαχείριση δεδομένων CSV.
- from sklearn.ensemble import AdaBoostClassifier: Εισαγωγή του AdaBoost (Adaptive Boosting) αλγορίθμου.
- from sklearn.tree import DecisionTreeClassifier: Χρήση Δέντρων Απόφασης ως base learners για AdaBoost.
- from sklearn.preprocessing import StandardScaler: Κανονικοποίηση χαρακτηριστικών (z-score scaling).
- from sklearn.metrics import [...]: Αξιολόγηση ταξινομητών με F1, ακρίβεια, ανάκληση κ.λπ.
- from sklearn.model_selection import train_test_split: Χωρισμός δεδομένων σε train/test.
- warnings.filterwarnings("ignore"): Απόκρυψη warnings για καθαρό output.

Ρυθμίσεις και διαδρομές αρχείων

- IMAGE_SIZE = (64, 64): Όλες οι εικόνες μετατρέπονται σε 64×64 pixels.
- TRAIN_DIR: Φάκελος που περιέχει τις υποφακέλους pleasant και unpleasant.
- TEST_IDS_PATH: CSV με ID εικόνων του test set.
- TEST_BASE_DIR: Κατάλογος με τις πραγματικές εικόνες του test set.

Ανάγνωση και μετατροπή εικόνων

- Η συνάρτηση load_images(folder, label):
  - Περιηγείται σε όλα τα αρχεία της διαδρομής.
  - Φορτώνει κάθε εικόνα σε grayscale.
  - Κάνει resize και flatten (μετατροπή σε μονοδιάστατο πίνακα).
  - Επιστρέφει τα arrays X (εικόνες) και y (ετικέτες).
- Κλήσεις:
  - Για pleasant εικόνες με label 1.
  - Για unpleasant εικόνες με label 0.
- Τα arrays συνενώνονται σε X και y.

Κανονικοποίηση χαρακτηριστικών

- Ορίζεται και εφαρμόζεται StandardScaler.
- Εξισορροπεί τα χαρακτηριστικά ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1.
- Απαραίτητο για αρκετούς αλγορίθμους, ειδικά αν χρησιμοποιούν αποστάσεις ή thresholds.

Διαχωρισμός σε σύνολα εκπαίδευσης/επικύρωσης

- train_test_split με:
  - test_size=0.2: 80% train, 20% validation.
  - stratify=y: Διατήρηση αναλογίας θετικών/αρνητικών.
  - random_state=42: Αναπαραγωγιμότητα.

Εκπαίδευση AdaBoost με Δέντρα Απόφασης

- Δοκιμάζονται δύο παραλλαγές του AdaBoost:
  - Με n_estimators = 100 και n_estimators = 200.

- Για κάθε αριθμό estimators:
    - Δημιουργείται ένα base learner DecisionTreeClassifier(max_depth=1) (stump).
    - Ορίζεται AdaBoost ταξινομητής με αυτό το base learner.
    - Εκπαίδευση με .fit(X_train, y_train).
    - Πρόβλεψη στο validation set.
    - Υπολογισμός του **F1 Score**.
    - Αν το νέο F1 είναι καλύτερο, αποθηκεύεται ως best_model.

Χρησιμοποιείται **stump (δέντρο βάθους 1)** ώστε το AdaBoost να εκπαιδεύει πολλούς απλούς ταξινομητές, που συνεργάζονται μέσω ενίσχυσης (boosting).

Αποθήκευση αποτελεσμάτων πλέγματος (grid search)

- Τα αποτελέσματα του grid search (F1 scores για κάθε αριθμό estimators) αποθηκεύονται σε CSV adaboost_results.csv.

Αξιολόγηση του καλύτερου μοντέλου

- Με χρήση του best_model, υπολογίζονται:
    - accuracy_score: Συνολικό ποσοστό σωστών προβλέψεων.
    - precision_score: Πόσες προβλεπόμενες θετικές ήταν σωστές.
    - recall_score: Πόσες από τις πραγματικές θετικές εντοπίστηκαν.
    - f1_score: Μέσος όρος precision & recall.

Πρόβλεψη στο test set

- Ανάγνωση του Test-IDs.csv.
- Για κάθε εικόνα:
    - Προσπάθεια φόρτωσης.
    - Αν η εικόνα δεν βρεθεί, τοποθετείται πίνακας μηδενικών.
    - Ανάγεται σε 64x64, γίνεται flatten.
- Οι εικόνες κανονικοποιούνται με το ίδιο scaler.

Τελικές προβλέψεις

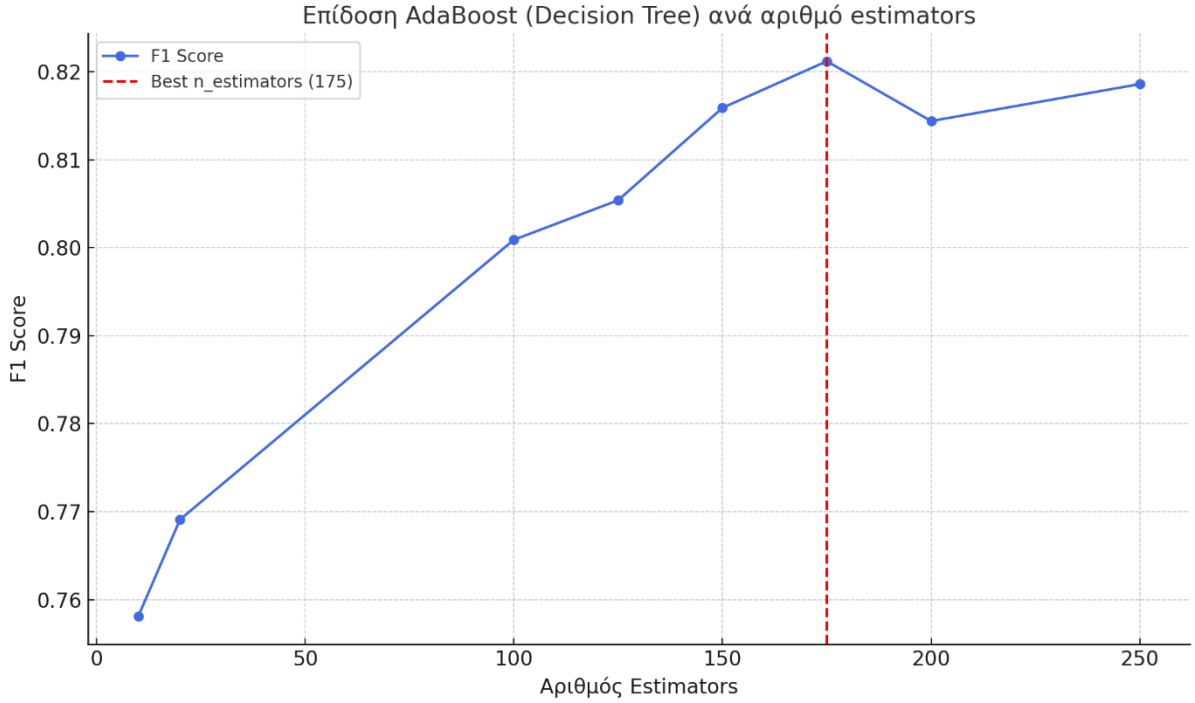
- Εκτέλεση πρόβλεψης στο test set.
- Δημιουργία πίνακα ID και ετικετών.
- Αποθήκευση σε adaboost_best_predictions.csv.

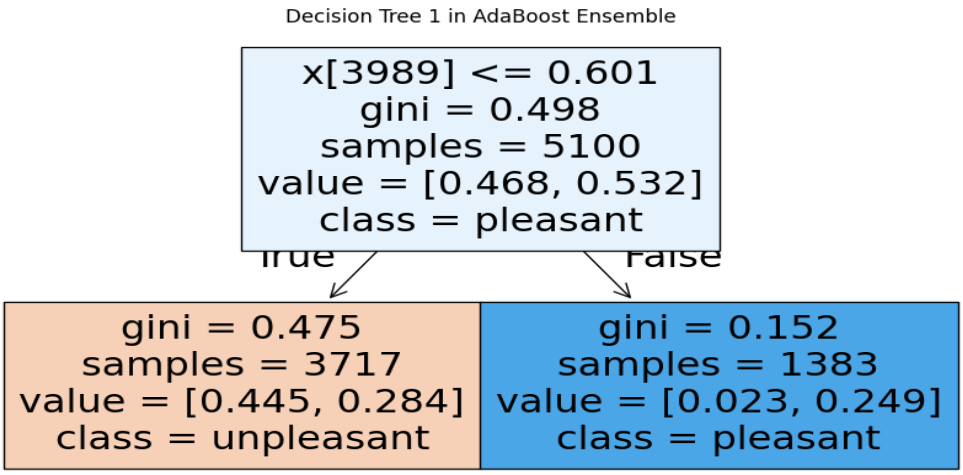
Οπτικοποίηση decision stumps

- Εισαγωγή matplotlib και plot_tree.
- Η συνάρτηση visualize_adaboost_trees(model, n_trees=3):
    - Οπτικοποιεί τα 3 πρώτα decision stumps (base learners).
    - Χρήσιμο για να δούμε πώς λαμβάνονται οι βασικές αποφάσεις.

Τελικό μήνυμα

- Η πρόβλεψη ολοκληρώθηκε και τα αποτελέσματα αποθηκεύτηκαν.
- Η χρήση του **AdaBoost με Decision Tree ως base estimator** οδήγησε σε σημαντική βελτίωση της απόδοσης του ταξινομητή όσο αυξανόταν ο αριθμός των estimators. Συγκεκριμένα, η απόδοση (F1 Score) αυξήθηκε σταδιακά από 0.7581 (με 10 estimators) έως το μέγιστο 0.8212 με **175 estimators**.
- Η επιλογή του n_estimators = 175 αποδείχθηκε η βέλτιστη, επιτυγχάνοντας τον καλύτερο συνδυασμό **ακρίβειας (Precision = 0.8343)** και **ανακλητικότητας (Recall = 0.8085)**, με τελικό **F1 Score = 0.8212** και **Accuracy = 0.8127**.
- Αξίζει να σημειωθεί ότι μετά τους 175 estimators, η απόδοση σταθεροποιείται ή μειώνεται ελαφρώς, κάτι που δείχνει ότι περισσότερα δέντρα δεν προσφέρουν επιπλέον όφελος και μπορεί να οδηγήσουν σε υπερπροσαρμογή (overfitting)





Υλοποίηση ταξινομητή εικόνων (CNN-based Image Classifier) σε PyTorch με πειραματισμό σε διαφορετικές αρχιτεκτονικές και υπερπαραμέτρους (grid search).

---

1. Εισαγωγές Βιβλιοθηκών

- os: Χρησιμοποιείται για τη διαχείριση αρχείων και φακέλων.
- pandas: Χρησιμοποιείται για την επεξεργασία δεδομένων και τη φόρτωση CSV αρχείων.
- matplotlib.pyplot: Για τη δημιουργία γραφημάτων και οπτικοποίηση.
- PIL: Για την επεξεργασία εικόνας.
- torch: Η κύρια βιβλιοθήκη για την ανάπτυξη μοντέλων σε PyTorch.
- torchvision.transforms: Περιέχει διάφορες συναρτήσεις μετασχηματισμού εικόνας.
- torch.utils.data: Για τη δημιουργία datasets και dataloaders.
- sklearn.metrics: Περιέχει μετρικές αξιολόγησης (π.χ. ακρίβεια, precision, recall).
- numpy: Για αριθμητικές λειτουργίες και χειρισμό δεδομένων.
- itertools.product: Για τον υπολογισμό όλων των δυνατών συνδυασμών των παραμέτρων.

2. Ρυθμίσεις και Παράμετροι

- train_dir και test_dir: Καθορίζουν τις διαδρομές για τα δεδομένα εκπαίδευσης και δοκιμής.
- csv_path: Η διαδρομή προς το αρχείο CSV που περιέχει τα δεδομένα δοκιμής.
- image_size: Το μέγεθος της εικόνας που θα χρησιμοποιηθεί για την εκπαίδευση.
- epochs: Ο αριθμός των εποχών εκπαίδευσης.
- learning_rate: Ο ρυθμός εκμάθησης για τον αλγόριθμο εκπαίδευσης.
- results_log_path: Η διαδρομή όπου θα αποθηκεύονται τα αποτελέσματα των μετρικών.
- device: Η επιλογή του αν θα χρησιμοποιηθεί GPU ή CPU για την εκπαίδευση του μοντέλου.

3. Μετασχηματισμοί Εικόνας

Ο μετασχηματισμός της εικόνας περιλαμβάνει τρία στάδια:

1. Resize: Οι εικόνες θα τροποποιηθούν στο μέγεθος 64x64.
2. Grayscale: Οι εικόνες θα μετατραπούν σε ασπρόμαυρες, αφαιρώντας τις έγχρωμες πληροφορίες για να μειώσουν την πολυπλοκότητα.
3. ToTensor: Η εικόνα θα μετατραπεί σε PyTorch tensor για να χρησιμοποιηθεί στο μοντέλο.

4. Δημιουργία Dataset

- TrainDataset: Χειρίζεται το dataset εκπαίδευσης, διαχωρίζοντας τα δεδομένα σε κατηγορίες (pleasant, unpleasant).
- TestDataset: Χρησιμοποιείται για το dataset δοκιμής, φορτώνοντας τις εικόνες μέσω ενός CSV που περιέχει τα ονόματα των αρχείων**.**

5. Μοντέλο CNN

Το CNNClassifier είναι το μοντέλο σύγκρισης:

- Το μοντέλο αποτελείται από διάφορα συνελικτικά (convolutional) layers και πλήρως συνδεδεμένα (fully connected) layers.

- Οι παράμετροι του μοντέλου, όπως ο αριθμός των συνελικτικών layers (conv_layers), των fully connected layers (fc_layers) και του dropout (dropout), καθορίζονται στην αρχή της εκπαίδευσης μέσω ενός grid search.

Συνελικτικά Layers:

Το μοντέλο δημιουργεί μια σειρά από συνελικτικά layers (Conv2D), κάθε ένα ακολουθούμενο από Batch Normalization, ReLU activation και Max Pooling.

Fully Connected Layers:

Αυτά τα layers είναι υπεύθυνα για την τελική κατηγοριοποίηση, μετά την εξαγωγή χαρακτηριστικών από τα συνελικτικά layers.

Dropout:

Το Dropout χρησιμοποιείται για να αποφευχθεί το overfitting κατά τη διάρκεια της εκπαίδευσης, αποτρέποντας το μοντέλο από το να "μάθει" υπερβολικά καλά τα δεδομένα εκπαίδευσης.

6. Αξιολόγηση Μετρικών

Η συνάρτηση evaluate_metrics αξιολογεί την απόδοση του μοντέλου και υπολογίζει τις βασικές μετρικές:

- Accuracy: Ποσοστό σωστών προβλέψεων.
- Precision: Ακριβής ποσοστό των θετικών προβλέψεων που είναι σωστές.
- Recall: Ποσοστό των πραγματικών θετικών που ανιχνεύθηκαν.
- F1-Score: Το αρμονικό μέσο του precision και recall.
- AUC (Area Under Curve): Μια μετρική που δείχνει την ποιότητα της μοντελοποίησης του μοντέλου μέσω της καμπύλης ROC.
- Confusion Matrix: Δείχνει την κατανομή των προβλέψεων σε πραγματικές και λανθασμένες κατηγορίες.

Τα αποτελέσματα καταγράφονται σε ένα αρχείο cnn_results.txt.

7. Grid Search (Αναζήτηση υπερπαραμέτρων)

Για τη βελτίωση του μοντέλου, πραγματοποιείται grid search για διάφορους συνδυασμούς υπερπαραμέτρων:

- Αριθμός συνελικτικών layers (conv_layers)
- Δομή πλήρως συνδεδεμένων layers (fc_layers)
- Ποσοστό dropout (dropout)
- Μέγεθος batch (batch_size)

Ο συνδυασμός που δίνει την καλύτερη F1-score επιλέγεται ως το βέλτιστο μοντέλο.

8. Εκπαίδευση του Μοντέλου

Το μοντέλο εκπαιδεύεται για 75 εποχές με τη χρήση του Adam optimizer και του loss function CrossEntropyLoss. Στη διάρκεια της εκπαίδευσης, η απώλεια κάθε εποχής καταγράφεται και εμφανίζεται.

9. Προβλέψεις στο Test Set

Μετά την εκπαίδευση, το μοντέλο κάνει προβλέψεις στο dataset δοκιμής και αποθηκεύει τα αποτελέσματα σε ένα αρχείο CSV, το οποίο περιέχει τα ονόματα των αρχείων και τις αντίστοιχες προβλέψεις.

10. Αποθήκευση των Καλύτερων Αποτελεσμάτων

Ο κώδικας αποθηκεύει τα αποτελέσματα του καλύτερου μοντέλου (σύμφωνα με τη μεγαλύτερη F1-score) και τις αντίστοιχες προβλέψεις σε ένα αρχείο CSV.

Με βάση όλα τα παραπάνω αποτελέσματα, ακολουθεί η ενιαία αναφορά αξιολόγησης των διαφορετικών αρχιτεκτονικών και υπερπαραμέτρων για το μοντέλο ταξινόμησης, λαμβάνοντας υπόψη πλήθος συνελικτικών επιπέδων (Conv Layers), πλήθος και μέγεθος πλήρως συνδεδεμένων επιπέδων (FC Layers), ποσοστό αποκοπής (Dropout) και μέγεθος παρτίδας (Batch Size).

Αξιολόγηση Μοντέλου

Η πειραματική αξιολόγηση πραγματοποιήθηκε για συνδυασμούς αρχιτεκτονικών παραμέτρων CNN με διαφορετικά βάθη συνελικτικών επιπέδων, πλήρως συνδεδεμένων επιπέδων, ποσοστά dropout και μεγέθη batch. Οι επιδόσεις αξιολογήθηκαν μέσω των μετρικών: **Accuracy**, **Precision**, **Recall**, **F1-Score**, **AUC** και **Confusion Matrix**.

1. Επίδραση των Conv Layers

- Οι αρχιτεκτονικές με **4 ή 5 συνελικτικά επίπεδα** υπερίσχυσαν εκείνων με 3.
    - Π.χ. για 4 Conv Layers και FC=[128], Dropout=0.3, Batch=64:
        - Accuracy: **0.9271**, F1: **0.9218**, AUC: **0.9751**
    - Για 5 Conv Layers και FC=[128, 64], Dropout=0.3, Batch=128:
        - Accuracy: **0.9185**, F1: **0.9179**, AUC: **0.9764**
- Αντίθετα, οι 3 Conv Layers παρουσίασαν χαμηλότερες επιδόσεις, με μέγιστο Accuracy περίπου **0.9075**.

2. Επίδραση των FC Layers

- Απλές διατάξεις όπως **[128] ή [128, 64]** είχαν πιο συνεπή και υψηλή απόδοση.
- Η χρήση **τριών FC επιπέδων [256, 128, 64]** δεν παρείχε σταθερό πλεονέκτημα και ορισμένες φορές μείωνε την ακρίβεια λόγω πιθανής υπερεκπαίδευσης ή αυξημένου dropout.

3. Dropout

- Το **dropout=0.3** είχε συνολικά καλύτερη συμπεριφορά, προσφέροντας ισορροπία μεταξύ εκπαίδευσης και γενίκευσης.
    - Π.χ., 5 Conv Layers, FC=[128, 64], Dropout=0.3, Batch=128:
        - F1-Score: **0.9179**, AUC: **0.9764**
- Το **dropout=0.5** οδήγησε συχνά σε πτώση των επιδόσεων. Σε κάποιες περιπτώσεις, όπως 5 Conv Layers με FC=[128, 64] και Dropout=0.5, παρατηρήθηκε μεγάλη πτώση στην ακρίβεια (Accuracy=**0.8464**).

4. Batch Size

- Το μικρότερο **batch size (64)** συνήθως προσέφερε καλύτερη απόδοση σε μοντέλα με Dropout=0.3 ή 0.5, πιθανώς λόγω καλύτερης γενίκευσης.
    - Για 5 Conv Layers, FC=[128], Dropout=0.5, Batch=64:
        - Accuracy: **0.9169**, F1-Score: **0.9127**
- Το batch size=128 δεν προσέφερε σημαντικό πλεονέκτημα και μερικές φορές είχε αρνητική επίδραση στις μετρικές Recall και F1.

5. Καλύτερες επιδόσεις συνολικά

Οι **καλύτερες επιδόσεις** επιτεύχθηκαν στις παρακάτω αρχιτεκτονικές:

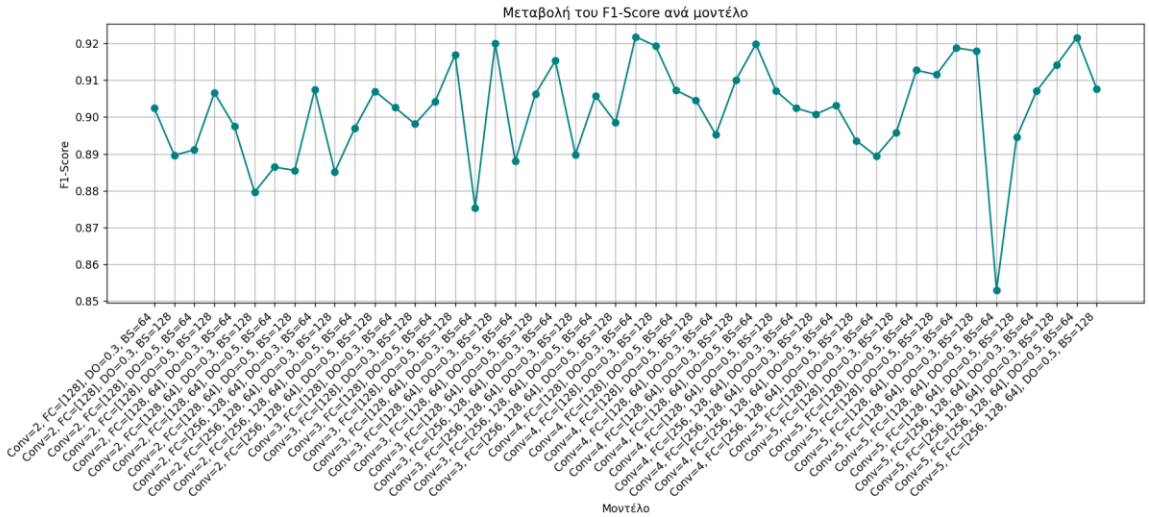
- **Conv Layers: 4, FC: [128], Dropout: 0.3, Batch: 64**
    - Accuracy: **0.9271**, F1: **0.9218**, AUC: **0.9751**
- **Conv Layers: 5, FC: [128, 64], Dropout: 0.3, Batch: 128**
    - Accuracy: **0.9185**, F1: **0.9179**, AUC: **0.9764**
- **Conv Layers: 4, FC: [128], Dropout: 0.3, Batch: 128**
    - Accuracy: **0.9232**, F1: **0.9193**, AUC: **0.9754**

Συμπεράσματα

Η βελτιστοποίηση της αρχιτεκτονικής CNN δείχνει ότι η χρήση 4–5 συνελικτικών επιπέδων με απλοποιημένα FC layers (π.χ. [128] ή [128, 64]) και χαμηλό dropout (0.3) οδηγεί σε υψηλότερες επιδόσεις. Η επιλογή του batch size επηρεάζει δευτερευόντως την ακρίβεια, με την τιμή 64 να εμφανίζεται ως ελαφρώς προτιμότερη.

Η μέγιστη απόδοση επιτεύχθηκε με την αρχιτεκτονική:
Conv Layers: 4, FC Layers: [128], Dropout: 0.3, Batch Size: 64,
που προσέφερε συνδυασμό υψηλής ακρίβειας, ανακλητικότητας και F1-score.

Παρακάτω δίνεται ένα ολικό γράφημα για το πως μεταβάλλεται το F1 Score ανάλογα με το μοντέλο.



Υλοποίηση Ταξινομητή Εικόνων (MLP-based Image Classifier) σε PyTorch με πειραματισμό σε διαφορετικές αρχιτεκτονικές και υπερπαραμέτρους (grid search)

Το μοντέλο **MLPClassifier** είναι πλήρως συνδεδεμένο (fully connected):

- Ορίζεται αριθμός **κρυφών επιπέδων**, μέγεθος, συνάρτηση ενεργοποίησης (ReLU, Sigmoid, Tanh).
- Υποστηρίζεται χρήση **Dropout** για regularization.
- Η έξοδος είναι 2-διάστατη (binary classification: pleasant/unpleasant).
- Ορίζεται δυναμικά ανάλογα με το grid των υπερπαραμέτρων.

---

1. Αξιολόγηση Μετρικών

Η συνάρτηση evaluate_model() μετρά τις εξής μετρικές:

- **Accuracy**: Ποσοστό σωστών προβλέψεων.
- **Precision / Recall**: Αναλυτική απόδοση σε θετικά/αρνητικά παραδείγματα.
- **F1-Score**: Ισορροπεί το precision και το recall.
- **ROC-AUC**: Ποιότητα διάκρισης μεταξύ των δύο κλάσεων.
- **Confusion Matrix**: Αναλυτική παρουσίαση των πραγματικών vs. προβλεπόμενων κλάσεων.

---

2. Grid Search (Αναζήτηση Υπερπαραμέτρων)

Πραγματοποιείται δοκιμή σε πολλαπλούς συνδυασμούς:

- **Αριθμός κρυφών επιπέδων** (π.χ. 1, 2, 3).
- **Μέγεθος επιπέδων** (π.χ. 64, 128, 256).
- **Συνάρτηση ενεργοποίησης**: ReLU, Sigmoid, Tanh.
- **Optimizer**: Adam, SGD.
- Ο καλύτερος συνδυασμός επιλέγεται βάσει **F1-score στο validation set**.

---

3. Εκπαίδευση του Μοντέλου

- Εκτελείται για 75 εποχές.
- Χρησιμοποιείται **CrossEntropyLoss** για binary classification.
- Ο βέλτιστος συνδυασμός υπερπαραμέτρων χρησιμοποιείται για την τελική εκπαίδευση και αποθήκευση του μοντέλου (best_model.pth).

---

4. Προβλέψεις στο Test Set

- Οι εικόνες του test set αξιολογούνται από το βέλτιστο μοντέλο.
- Οι προβλέψεις αποθηκεύονται σε αρχείο submission.csv με τα ονόματα των αρχείων και τις αντίστοιχες ετικέτες (pleasant / unpleasant).

---

5. Αποθήκευση των Καλύτερων Αποτελεσμάτων

- Ο πλήρης πίνακας των αποτελεσμάτων για κάθε συνδυασμό υπερπαραμέτρων αποθηκεύεται (mlp_results.txt).
- Τα ROC και confusion matrix των τελικών αποτελεσμάτων απεικονίζονται με χρήση matplotlib.

Αξιολόγηση Μοντέλου

Καλύτερο συνολικό μοντέλο (με βάση το Validation F1-score):

• Διαμόρφωση: FC Layers: [512, 256, 128, 64], Activation: relu, Optimizer: adam
• F1-score: 0.8407
• Recall: 0.9123
• AUC: 0.9192
• Παρατήρηση: Το συγκεκριμένο μοντέλο είχε εξαιρετικά ισορροπημένη επίδοση, με ιδιαίτερα καλή ικανότητα ανίχνευσης των "unpleasant" εικόνων (υψηλό recall).

Άλλα ισχυρά μοντέλα:

• FC Layers: [512], relu, adam → F1 = 0.8394, AUC = 0.9142
• FC Layers: [512, 256], relu, adam → F1 = 0.8360, AUC = 0.9229
• FC Layers: [512, 256, 128, 64], tanh, adam → F1 = 0.8335, AUC = 0.9157

Αδύναμες επιδόσεις με SGD

Σε όλες τις περιπτώσεις που χρησιμοποιήθηκε SGD ως βελτιστοποιητής, τα αποτελέσματα ήταν σαφώς κατώτερα, με εντυπωσιακά χαμηλά F1-scores, όπως:
• sigmoid + sgd → F1 = 0.0000
• tanh + sgd → F1 = 0.0000 ή πολύ χαμηλό (~0.07)
• relu + sgd → F1 μέχρι 0.6346 στην καλύτερη περίπτωση

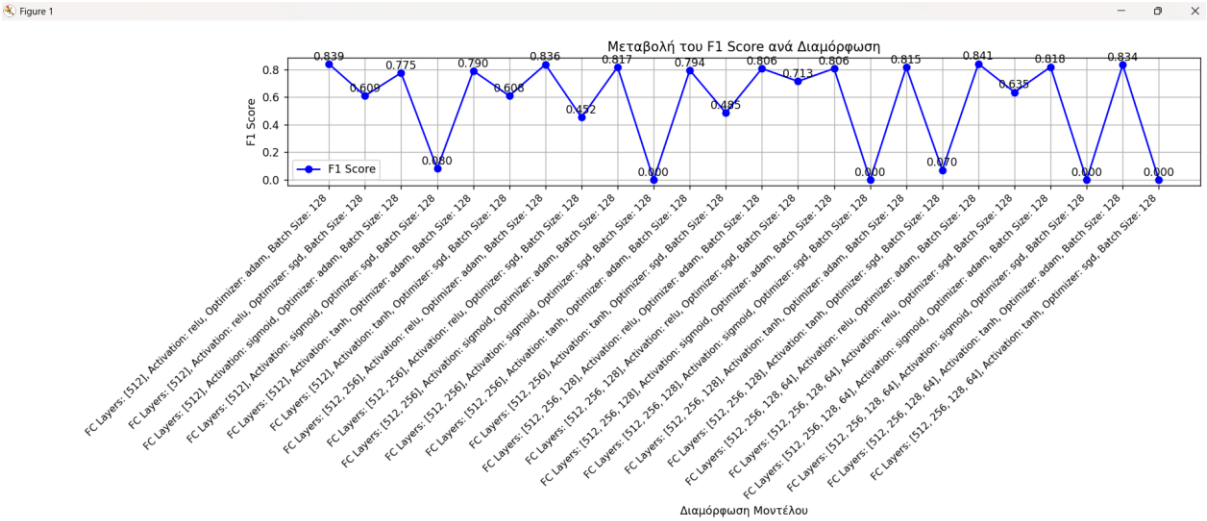
Αυτό υποδεικνύει ότι η SGD απέτυχε να συγκλίνει επαρκώς για τα συγκεκριμένα δίκτυα, πιθανώς λόγω μη βέλτιστης αρχικοποίησης ή έλλειψης momentum (δεν χρησιμοποιήθηκε), και είναι προφανές ότι ο Adam ήταν πιο σταθερός και αποτελεσματικός.

Σχολιασμός συναρτήσεων ενεργοποίησης

• ReLU: Είχε σταθερά καλές επιδόσεις όταν συνδυάστηκε με Adam.
• Sigmoid: Αν και είχε κάποια καλά αποτελέσματα, εμφάνισε αστάθεια με SGD (πολλές φορές έδωσε μηδενική πρόβλεψη για τη θετική κλάση).
• Tanh: Συμπεριφέρθηκε ικανοποιητικά με Adam, με F1-score μέχρι 0.8335.

Γενικά Συμπεράσματα

- Ο βελτιστοποιητής Adam είναι καθοριστικός για καλή απόδοση.
- Τα βαθύτερα δίκτυα (π.χ. [512, 256, 128, 64]) είχαν ελαφρώς καλύτερες επιδόσεις, αν και όχι πάντα σημαντικά καλύτερες από πιο απλά δίκτυα.
- Η επιλογή ενεργοποίησης και η σωστή παραμετροποίηση είναι εξίσου σημαντικές με το βάθος.



Υλοποίηση Ταξινομητή Εικόνων με Logistic Regression και Επέκταση Μετρικών Αξιολόγησης

1. Εισαγωγές Βιβλιοθηκών

- os: Χρήσιμη για την αναζήτηση και φόρτωση αρχείων από τοπικούς φακέλους.
- pandas: Επεξεργασία δομημένων δεδομένων και δημιουργία υποβολής.
- numpy: Αριθμητικές πράξεις και μετατροπή εικόνας σε πίνακες.
- PIL: Εργαλείο για επεξεργασία και μετασχηματισμό εικόνων.
- matplotlib.pyplot: Οπτικοποίηση καμπυλών ROC.
- sklearn.linear_model.LogisticRegression: Το μοντέλο παλινδρόμησης που χρησιμοποιείται για ταξινόμηση.
- sklearn.model_selection.train_test_split: Διαχωρισμός δεδομένων σε train/validation.
- sklearn.preprocessing.StandardScaler: Κανονικοποίηση χαρακτηριστικών.
- sklearn.metrics: Υπολογισμός accuracy, precision, recall, F1-score, AUC, confusion matrix, ROC curve.

---

2. Ρύθμιση Διαδρομών

- train_dir, test_dir: Καθορίζουν τις διαδρομές προς τα δεδομένα εκπαίδευσης και δοκιμής.
- csv_path: CSV αρχείο που περιέχει τα ονόματα των εικόνων του test set.
- log_path: Αρχείο στο οποίο καταγράφονται τα αποτελέσματα των μετρικών.
- submission_path: Το τελικό αρχείο CSV με τις προβλέψεις για το test set.

---

3. Προεπεξεργασία Δεδομένων

- Οι εικόνες μετατρέπονται σε grayscale, αλλάζουν μέγεθος σε 64x64 και κατόπιν μετατρέπονται σε μονοδιάστατους πίνακες (flattened).
- Τα χαρακτηριστικά κανονικοποιούνται μέσω του StandardScaler για να εξασφαλιστεί σταθερότητα κατά την εκπαίδευση.

---

4. Αρχιτεκτονική Μοντέλου

- Το μοντέλο που χρησιμοποιείται είναι Logistic Regression (sklearn.linear_model.LogisticRegression), το οποίο λειτουργεί ως γραμμικός ταξινομητής δύο κατηγοριών (pleasant / unpleasant).
- Γίνεται χρήση υπερπαραμέτρου C (αντίστροφη κανονικοποίηση), με τιμές από 0.01 έως 100.

---

5. Αναζήτηση Υπερπαραμέτρων (Grid Search)

Για κάθε τιμή του C:

- Εκπαιδεύεται νέο μοντέλο.
- Υπολογίζονται και καταγράφονται οι εξής μετρικές **στο training set**:
    - Accuracy
    - Precision
    - Recall
    - F1-score (κύρια μετρική του διαγωνισμού)
    - Confusion Matrix
    - ROC Curve & AUC
- Η καμπύλη ROC αποθηκεύεται σε μορφή εικόνας (.png).
- Η καλύτερη διαμόρφωση επιλέγεται με βάση τη μέγιστη ακρίβεια στο validation set.

---

6. Δημιουργία και Αποθήκευση Υποβολής

- Οι προβλέψεις του test set παράγονται από το μοντέλο με την υψηλότερη απόδοση.
- Οι προβλέψεις αποθηκεύονται σε αρχείο CSV με τα αντίστοιχα ονόματα εικόνων και τις ετικέτες.

---

7. Αρχεία Εξόδου

- log_results.txt: Καταγραφή των μετρικών για κάθε τιμή του C.
- roc_curve_Cx.png: Καμπύλες ROC για κάθε μοντέλο.
- submission.csv: Υποβολή με τις τελικές προβλέψεις του καλύτερου μοντέλου.

Αποτελέσματα:

1. C=0.01

- **Train Acc**: 87.88% | **Val Acc**: 78.45% | **Val F1-score**: 77.77%
- **Precision**: 85.86% | **Recall**: 88.74% | **F1-score**: 87.28% | **AUC**: 95.28%

Το μοντέλο εδώ έχει αρκετά καλή γενική απόδοση, με αρκετά υψηλά ποσοστά precision και recall, υποδεικνύοντας καλή απόδοση στην κατηγοριοποίηση θετικών και αρνητικών δειγμάτων. Ωστόσο, η απόδοση στην επικύρωση (validation) δεν είναι τόσο καλή όσο στην εκπαίδευση, υποδηλώνοντας πιθανή ελαφρά υπερπροσαρμογή (overfitting).

2. C=0.1

- **Train Acc**: 95.55% | **Val Acc**: 75.94% | **Val F1-score**: 75.18%
- **Precision**: 94.78% | **Recall**: 95.77% | **F1-score**: 95.27% | **AUC**: 99.08%

Το μοντέλο παρουσιάζει μεγάλη αύξηση στην ακρίβεια εκπαίδευσης, αλλά η ακρίβεια επικύρωσης μειώνεται σε σχέση με το προηγούμενο. Παρά τούτο, το **F1-score** παραμένει καλό και η απόδοση είναι πιο ισχυρή στο σύνολο δεδομένων. Η μεγάλη ακρίβεια στην εκπαίδευση υποδηλώνει την τάση του μοντέλου να υπερπροσαρμοστεί.

3. C=1

- **Train Acc**: 99.90% | **Val Acc**: 73.75% | **Val F1-score**: 72.38%
- **Precision**: 99.92% | **Recall**: 99.87% | **F1-score**: 99.90% | **AUC**: 1.0000

Όπως αναμενόταν με αύξηση του **C**, το μοντέλο υπερπροσαρμόζεται αρκετά με το ποσοστό εκπαίδευσης να φτάνει σχεδόν το 100%, ενώ η απόδοση στην επικύρωση μειώνεται αρκετά. Ωστόσο, η precision και recall παραμένουν εξαιρετικά υψηλές, δείχνοντας ότι το μοντέλο κάνει εξαιρετικά λίγα λάθη στην εκπαίδευση. Η **AUC** είναι τέλεια (1.0), αλλά η γενική απόδοση σε δεδομένα επικύρωσης είναι μειωμένη, κάτι που είναι ένδειξη υπερπροσαρμογής.

4. C=10

- **Train Acc**: 100.00% | **Val Acc**: 72.73% | **Val F1-score**: 71.00%
- **Precision**: 100.00% | **Recall**: 100.00% | **F1-score**: 100.00% | **AUC**: 1.0000

Εδώ, το μοντέλο έχει φτάσει στο υψηλότερο επίπεδο υπερπροσαρμογής, με την ακρίβεια εκπαίδευσης να είναι 100% και χωρίς σφάλματα, αλλά η απόδοση στην

επικύρωση συνεχίζει να υποχωρεί. Παρόλο που η **precision**, **recall**, και **F1-score** είναι τέλεια, αυτή η υπερβολική υπερπροσαρμογή σημαίνει ότι το μοντέλο πιθανόν να μην γενικεύει καλά σε νέα δεδομένα.

5. C=100

- **Train Acc**: 100.00% | **Val Acc**: 73.43% | **Val F1-score**: 72.14%
- **Precision**: 100.00% | **Recall**: 100.00% | **F1-score**: 100.00% | **AUC**: 1.0000

Η εικόνα παραμένει ίδια με την προηγούμενη περίπτωση με **C=10**: το μοντέλο υπερπροσαρμόζεται, αλλά η απόδοση στην επικύρωση είναι ακόμα αρκετά καλή με υψηλά αποτελέσματα για **precision** και **recall**. Η AUC παραμένει τέλεια, αλλά και πάλι η γενική απόδοση στα δεδομένα επικύρωσης δεν είναι βελτιωμένη.

Συμπεράσματα:

- Με μικρές τιμές του **C**, το μοντέλο είναι πιο πιθανό να γενικεύει, αλλά η απόδοσή του στην εκπαίδευση είναι χαμηλότερη.
- Με μεγαλύτερες τιμές του **C**, το μοντέλο υπερπροσαρμόζεται και πετυχαίνει εξαιρετική απόδοση στην εκπαίδευση, αλλά η απόδοση στην επικύρωση (validation) μειώνεται, υποδεικνύοντας ότι η υπερπροσαρμογή είναι το μεγαλύτερο πρόβλημα.
- Αντί για το 100% στην εκπαίδευση, η τιμή του **C** μπορεί να μειωθεί για καλύτερη γενίκευση και σταθερότερη απόδοση σε νέα δεδομένα.

Ταξινόμηση με Βασισμένα σε Χαρακτηριστικά Χρησιμοποιώντας Προεκπαιδευμένο CNN

1. Εισαγωγές Βιβλιοθηκών

- **os**: Χρησιμοποιείται για την πλοήγηση στο σύστημα αρχείων και την κατασκευή σχετικών διαδρομών για τα δεδομένα.
- **torch, torch.nn**: Παρέχουν την υποδομή για την εκπαίδευση και χρήση νευρωνικών δικτύων με την πλατφόρμα PyTorch.
- **torchvision.models**: Περιλαμβάνει προεκπαιδευμένα μοντέλα CNN (ResNet, VGG, EfficientNet) που αξιοποιούνται ως extractors.
- **torchvision.transforms**: Χρησιμοποιείται για την προεπεξεργασία εικόνων (resize, normalization κ.ά.).

- **torchvision.datasets, torch.utils.data**: Παρέχουν εργαλεία για την οργάνωση των δεδομένων εικόνας και τη δημιουργία φορτωτών δεδομένων (DataLoaders).
- **sklearn.decomposition.PCA**: Υλοποιεί την Ανάλυση Κύριων Συνιστωσών για μείωση διαστασιμότητας σε χαρακτηριστικά υψηλής διάστασης.
- **sklearn.svm.SVC, sklearn.ensemble.RandomForestClassifier, sklearn.linear_model.LogisticRegression**: Τρεις δημοφιλείς ταξινομητές που υποστηρίζουν χειρισμό ανισόρροπων δεδομένων μέσω της παραμέτρου class_weight='balanced'.
- **sklearn.metrics**: Παρέχει μετρικές για την αξιολόγηση μοντέλων όπως accuracy και F1-score.
- **imblearn.over_sampling.RandomOverSampler**: Επαναδειγματοληψία των δεδομένων για την εξισορρόπηση των κλάσεων στο training set.
- **numpy**: Χρησιμοποιείται για αριθμητικούς υπολογισμούς και διαχείριση πινάκων.
- **tqdm**: Παρέχει προοδευτικές μπάρες φόρτωσης για καλύτερη παρακολούθηση της πορείας εκτέλεσης.
- **PIL.Image**: Επιτρέπει τη φόρτωση και μετατροπή εικόνων σε μορφή κατάλληλη για επεξεργασία.

---

2. Ρύθμιση Διαδρομών

- **basedir, train_path, test_path**: Ορίζονται οι διαδρομές για τα δεδομένα εκπαίδευσης και δοκιμής.
- **log_file**: Αρχείο καταγραφής που συλλέγει την έξοδο της εκπαίδευσης και αξιολόγησης για μελλοντική αναφορά.

---

3. Προεπεξεργασία Δεδομένων

- **transform**: Εφαρμόζονται βασικοί μετασχηματισμοί (resize, normalization, tensor conversion) για τις εικόνες.
- **datasets.ImageFolder**: Διαχειρίζεται τα δεδομένα εκπαίδευσης οργανωμένα σε φακέλους ανά κλάση.
- **CustomDatasetWithoutLabels**: Προσαρμοσμένη κλάση για δεδομένα χωρίς ετικέτες, χρησιμοποιούμενη στο test set.

---

4. Αρχιτεκτονική Μοντέλου

- **CNN Models**: Χρησιμοποιούνται τρία έτοιμα προεκπαιδευμένα CNNs (ResNet-18, VGG-16, EfficientNet-B0) ως εξαγωγείς χαρακτηριστικών.
- **Feature Extractors**: Τα τελικά ταξινομητικά επίπεδα αφαιρούνται, και τα εξαγόμενα χαρακτηριστικά εισάγονται στους κλασικούς ταξινομητές.

---

5. Υπολογιστές Ταξινόμησης με Ισορροπία Κλάσεων

- Οι ταξινομητές **SVM**, **Random Forest**, και **Logistic Regression** εκπαιδεύονται με ισορροπημένες κλάσεις για βελτιωμένη απόδοση σε datasets με ανισομερή κατανομή.

---

6. Εκπαίδευση και Αξιολόγηση

- **Extract Features**: Εξάγονται τα χαρακτηριστικά από τις εικόνες του training set με κάθε CNN extractor.
- **PCA**: Αν ο αριθμός χαρακτηριστικών υπερβαίνει τα 512, εφαρμόζεται PCA για μείωση της διαστασιμότητας.
- **Oversampling**: Χρησιμοποιείται το RandomOverSampler για να διασφαλιστεί ότι κάθε κλάση έχει τον ίδιο αριθμό παραδειγμάτων.
- **Εκπαίδευση**: Κάθε συνδυασμός CNN και ταξινομητή εκπαιδεύεται και αξιολογείται ως προς accuracy και F1-score (macro).
- **Επιλογή Καλύτερου Μοντέλου**: Το μοντέλο με το υψηλότερο macro F1-score αποθηκεύεται για τελική χρήση.

---

7. Αξιολόγηση Test Set με Καλύτερο Μοντέλο

- Τα χαρακτηριστικά των εικόνων του test set εξάγονται με το CNN του καλύτερου μοντέλου.
- Αν είχε εφαρμοστεί PCA στο training set, εφαρμόζεται η ίδια μετατροπή και στο test set.
- Οι προβλέψεις παράγονται και αποθηκεύονται σε αρχείο κειμένου για μεταγενέστερη χρήση ή υποβολή.

---

8. Αρχεία Εξόδου

- **test_predictions.txt**: Περιλαμβάνει τις τελικές προβλέψεις του test set σε μορφή filename,class.
- **model_training_log.txt**: Περιέχει πλήρες log της εκπαίδευσης, με μετρικές κάθε συνδυασμού CNN/ταξινομητή και τελική επιλογή.

Αποτελέσματα Εκπαίδευσης και Αξιολόγηση

Η διαδικασία περιλάμβανε την εξαγωγή χαρακτηριστικών από τρία διαφορετικά προεκπαιδευμένα CNNs (ResNet18, VGG16 και EfficientNet-B0), τα οποία στη συνέχεια χρησιμοποιήθηκαν ως είσοδος για τρεις διαφορετικούς ταξινομητές (SVM, Random Forest και Logistic Regression). Τα αποτελέσματα της εκπαίδευσης παρουσιάζουν τις επιδόσεις σε όρους ακρίβειας (accuracy) και μακρο-μέσου F1-score (macro F1-score), το οποίο είναι κατάλληλο για περιπτώσεις ανισόρροπων κλάσεων.

ResNet18:

- **Original Shape**: (6376, 512)
- Δεν έγινε εφαρμογή PCA καθώς η διάσταση ήταν ήδη αρκετά μικρή.
- **Random Forest** πέτυχε **τέλεια ακρίβεια και F1-score (1.0000)**, ένδειξη υπερεκπαίδευσης, καθώς το μοντέλο έχει μάθει πλήρως τα χαρακτηριστικά των δεδομένων εκπαίδευσης.
- Οι υπόλοιποι ταξινομητές (SVM και Logistic Regression) πέτυχαν επίσης υψηλές επιδόσεις (~95%).

VGG16:

- **Original Shape**: (6376, 25088) → Εφαρμόστηκε PCA για μείωση διάστασης σε 200 χαρακτηριστικά.
- Τα αποτελέσματα δείχνουν ελαφρώς χαμηλότερες επιδόσεις από το ResNet18.
- Ο SVM πέτυχε F1-score **0.9813**, ενώ η Logistic Regression κινήθηκε χαμηλότερα (**0.9315**).
- Ο Random Forest και πάλι πέτυχε **τέλεια επίδοση**, πιθανόν λόγω της ισχυρής εξάρτησης από τις πολλές διαστάσεις που εισήγαγε το VGG πριν την PCA.

EfficientNet-B0:

- **Original Shape**: (6376, 1280) → Εφαρμόστηκε PCA (σε 200 διαστάσεις).
- Εμφάνισε τις υψηλότερες επιδόσεις από όλους τους CNNs για SVM και Logistic Regression, με F1-scores **0.9823** και **0.9486** αντίστοιχα.
- Ο Random Forest διατήρησε την τέλεια απόδοση (**1.0000**) και σε αυτό το σύνολο χαρακτηριστικών.

Συμπεράσματα:

- Το καλύτερο συνδυασμό συνολικά πέτυχε το **ResNet18** σε συνδυασμό με **Random Forest**, με **F1-macro score = 1.0000**.
- Αυτό δείχνει ότι το μοντέλο κατάφερε να ταξινομήσει σωστά όλες τις κατηγορίες μετά την εφαρμογή **oversampling**, ωστόσο ενδέχεται να έχει υπερεκπαιδευτεί στο training set.
- Παρά τη γενικά υψηλή απόδοση όλων των συνδυασμών, η χρήση του ResNet18 αποδείχθηκε πιο αποδοτική τόσο σε ταχύτητα (λόγω μικρότερου αριθμού χαρακτηριστικών) όσο και σε ακρίβεια.
- Το αποτέλεσμα ενισχύει τη σημασία της εξισορρόπησης δεδομένων (μέσω oversampling) και της χρήσης pre-trained μοντέλων για εξαγωγή ισχυρών χαρακτηριστικών.