Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

ΥΛΟΠΟΙΗΤΙΚΟ PROJECT 2025

ΑΛΕΞΑΚΗ ΒΑΣΙΛΙΚΗ ΑΜ: 1097464

ΞΗΝΤΑΡΟΠΟΥΛΟΥ ΑΣΗΜΙΝΑ ΑΜ: 1093452

ΠΕΡΙΕΧΟΜΕΝΑ

Ερώτημα 1

Αρχείο κώδικα: data_descr_stats.py
Αρχείο εισόδου: data.csv
Παραγόμενο αρχείο: data_analysis_2025-06-01_12-42-31.txt, correlation_pairs_over_0.4_2025-05-18_00-24-21.txt, διαγράμματα σε φακέλους.

Στο ερώτημα 1 έχουμε δημιουργήσει τον κώδικα στο αρχείο **data_descr_stats.py** , ο οποίος παράγει ένα **csv** (data_analysis_2025-06-01_12-42-31) με τις πληροφορίες για τις στήλες και τα βασικά συγκεντρωτικά στατιστικά μεγέθη για ολόκληρο το αρχείο **data.csv**. Έπειτα, δημιουργούμε τα παρακάτω διαγράμματα για οπτικοποίηση των δεδομένων μας.

- HISTOGRAMS
- HISTOGRAMS-LOGY (για καλύτερη απεικόνιση)
- BOXPLOTS
- HEATMAP-SELECTED COLS
- COUNTPLOTS
- VIOLIN PLOTS
- HEATMAP CORR>0.4

Όλα τα παραπάνω, **εκτός** από το **heatmap corrr>0.4** που γίνεται για **όλο** το σύνολο δεδομένων, γίνονται για **επιλεγμένες στήλες** (με βάση τα περιγραφικά στοιχεία) που έχει νόημα να τις αναλύσουμε.

Για αριθμητικά δεδομένα

```
selected_cols = [
    'Flow Duration', 'Fwd IAT Total', 'Flow IAT Max', 'Idle Max', 'Idle
Mean', 'Fwd IAT Min',
    'Fwd IAT Mean', 'Packet Length Std', 'Fwd Packet Length Max', 'Packet
Length Mean'
]
```

Για κατηγορικά δεδομένα (όλες οι κατηγορικές στήλες)

```
categorical_cols = ['Timestamp', 'Flow ID', 'Dst Port', 'Src IP', 'Label',
'Traffic Type', 'Traffic Subtype']
```

Βιβλιοθήκες

Για το ερώτημα χρησιμοποιούμε τις παρακάτω βιβλιοθήκες :

```
import matplotlib
matplotlib.use('TkAgg')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sys
import os
```

```
from datetime import datetime
import plotly.express as px
```

matplotlib : Η βασική βιβλιοθήκη για οπτικοποίηση γραφημάτων.
pandas : Η βασική βιβλιοθήκη για φόρτωση, επεξεργασία και ανάλυση πινάκων δεδομένων (π.χ για ανάγνωση του CSV αρχείου) .
numpy : Υποστηρίζει αριθμητικούς υπολογισμούς (π.χ. για τον έλεγχο τύπων δεδομένων).
matplotlib.pyplot : Παρέχει εύκολες συναρτήσεις για σχεδίαση γραφημάτων, όπως bar plots, line plots, histograms.
seaborn : Βιβλιοθήκη βασισμένη στη matplotlib για στατιστικά διαγράμματα.
sys : Καταγράφει τα print σε .txt αρχείο αντί για την κονσόλα.
os : Δείχνει το πλήρες path του αρχείου.
datetime : Δημιουργεί timestamp για το όνομα του αρχείου αναφοράς που αποθηκεύοουμε (π.χ., data_analysis_2025-06-06_13-42-00.txt).
plotly : Διαδραστική βιβλιοθήκη γραφημάτων.

Για να εγκαταστήσουμε τις παραπάνω βιβλιοθήκες χρειαστήκαμε την εντολή :

```
py -m pip install  +όνομα βιβλιοθήκης
```

Περιγραφικά - Στατιστικά στοιχεία

Επιστρέφει το **σχήμα** (γραμμές, στήλες) του πίνακα.

```
print("ΣΧΗΜΑ ΠΙΝΑΚΑ:", df.shape)
```

Τυπώνει τον **τύπο** κάθε στήλης (π.χ., int64, float64, object, datetime64).

```
print(df.dtypes)
```

Κατηγοριοποίηση **τύπων** (πόσες στήλες είναι αριθμητικές, κειμένου, ημερομηνίες κ.λπ.)

```
np.issubdtype(df[col].dtype, np.number)  # Αν είναι αριθμητική
np.issubdtype(df[col].dtype, np.datetime64)  # Αν είναι ημερομηνία
```

Πόσες **κενές τιμές** έχει κάθε στήλη.

```
df.isnull().sum()
```

Στατιστικά μόνο για **αριθμητικές** στήλες: **count, mean, std, min, 25%, 50%, 75%, max**

```
df.describe().transpose()
```

Στατιστικά για **όλες** τις στήλες -Περιλαμβάνει επίσης **unique, top, freq** για κειμενικές στήλες.

```
df.describe(include='all').transpose()
```

Εκτυπώνει όλα τα **ονόματα** των στηλών.

```
print(df.columns)
```

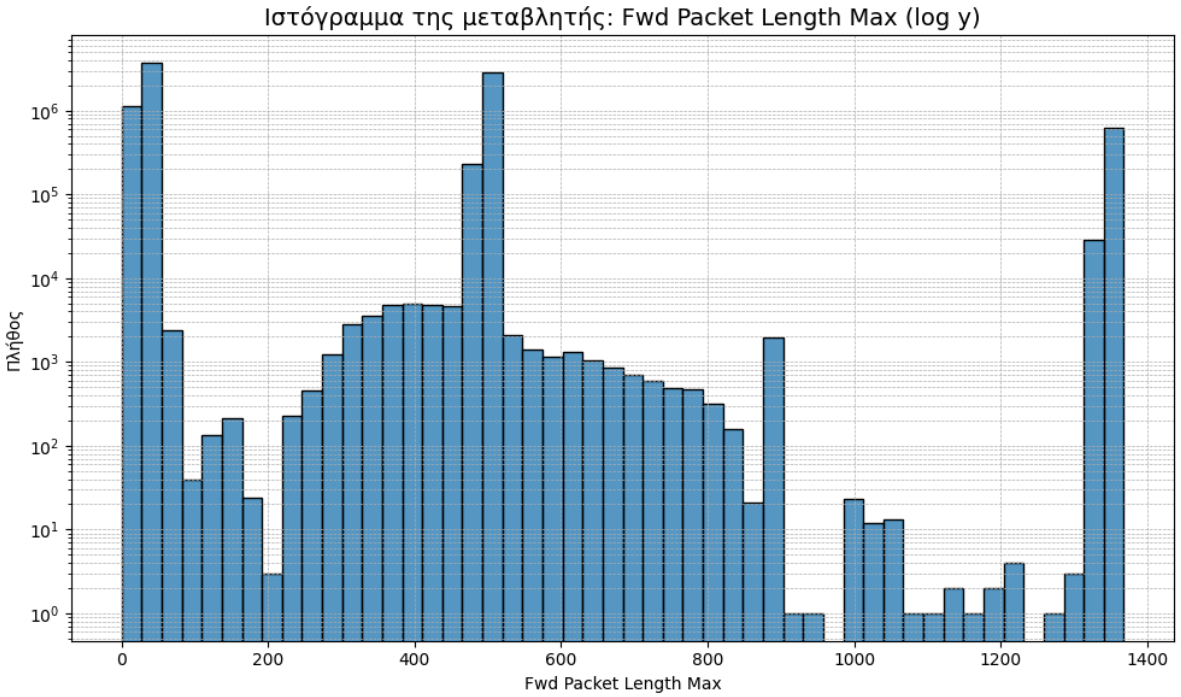
Γραφικές παραστάσεις

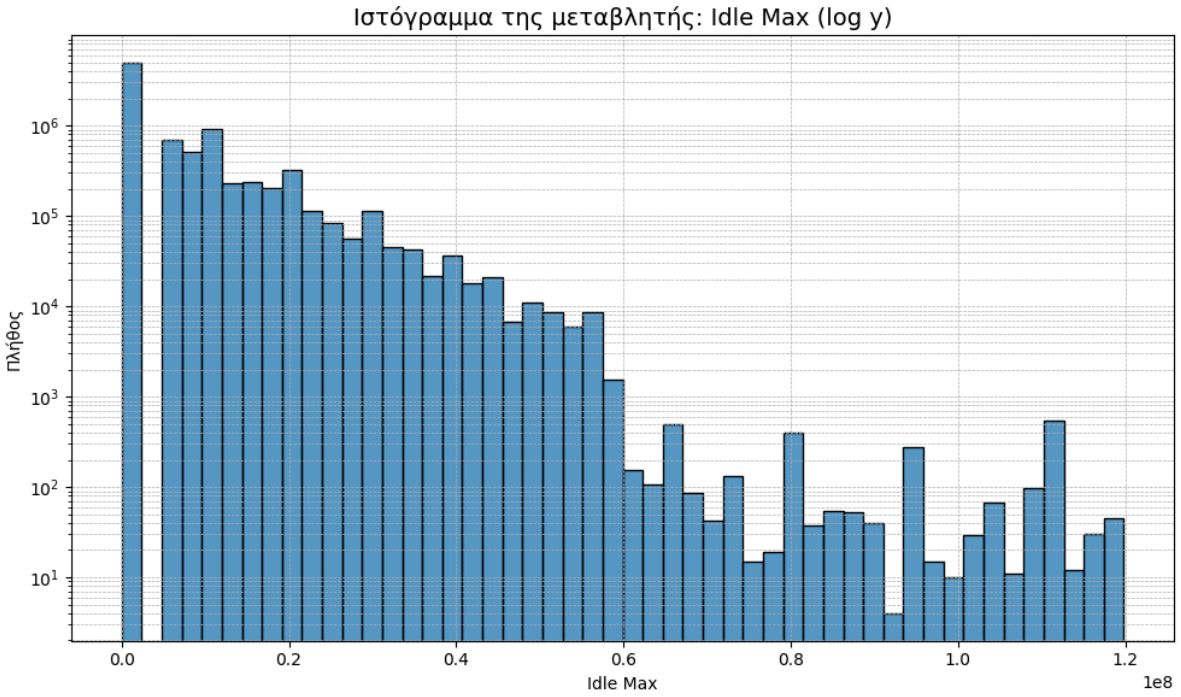
Histograms

Τα **ιστογράμματα** χρησιμοποιήθηκαν για να μελετηθεί η κατανομή των αριθμητικών μεταβλητών στο σύνολο δεδομένων. Με αυτά μπορούμε να εντοπίσουμε:

- αν μια μεταβλητή είναι κανονικά κατανεμημένη ή όχι
- αν υπάρχουν ασυμμετρίες ή ακραίες τιμές (outliers)
- ποιες τιμές εμφανίζονται πιο συχνά

Τα ιστογράμματα (histograms) εφαρμόστηκαν σε αριθμητικά δεδομένα τα οποία είχαν **μη μηδενική τυπική απόκλιση** (std > 0) και **αρκετές διαφορετικές τιμές** (unique > 10).

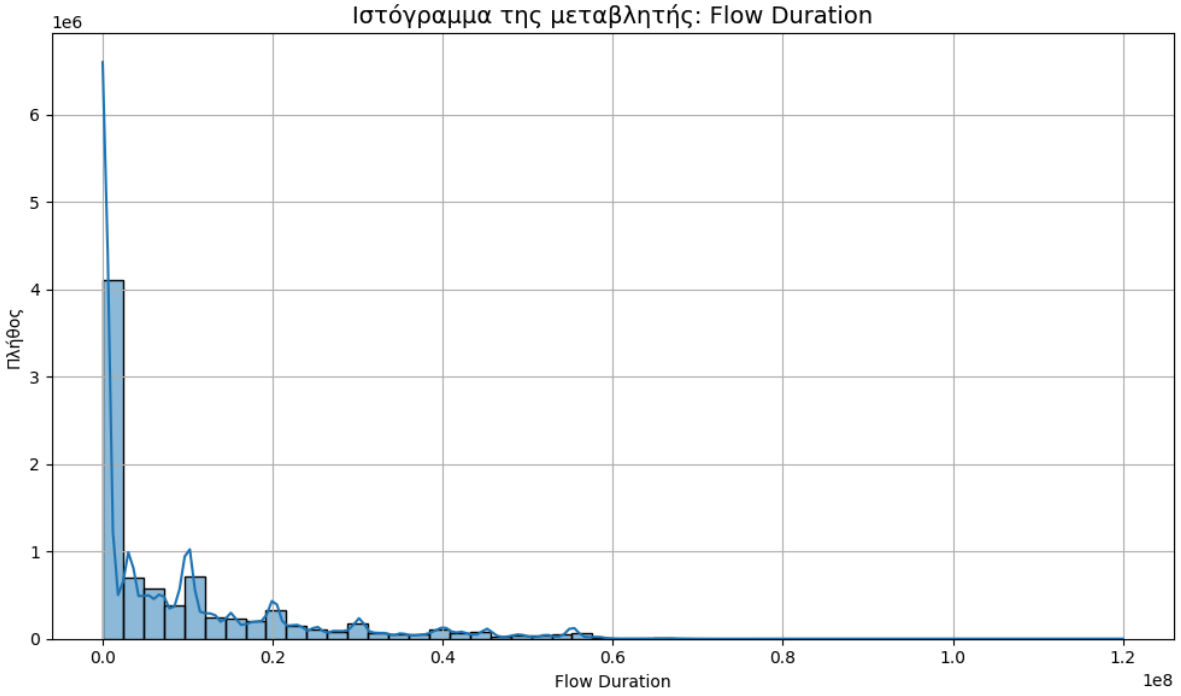




Για τα ιστογράμματα έχουμε εφαρμόσει 2 τεχνικές καθώς η συνηθισμένη λόγω των ανισόρροπων δεδομένων δεν αποδίδει στο μέγιστο.

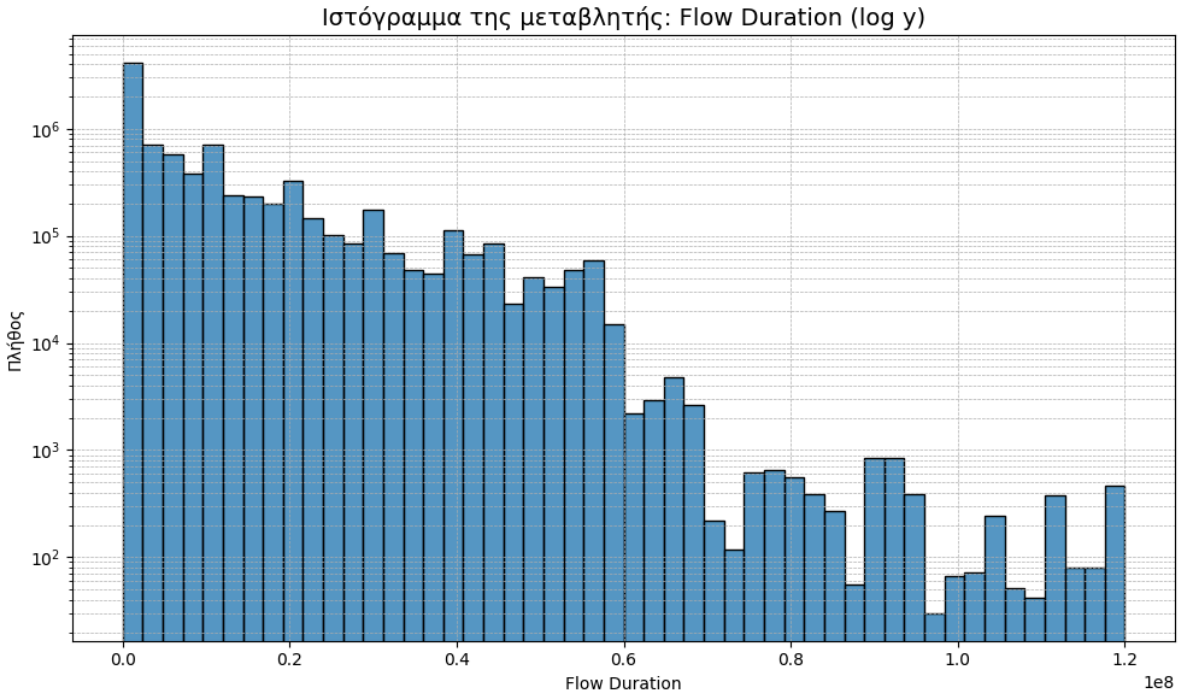
```
sns.histplot(data=df, x=col, kde=True, bins=50)
```

bins=50: Χωρίζει τα δεδομένα σε **50** διαστήματα (κάδους).
kde=True: Εμφανίζει και την **καμπύλη πυκνότητας** (Kernel Density Estimate), που δείχνει πιο "ομαλά" την κατανομή.



```
sns.histplot(data=df, x=col, bins=50)
plt.yscale('log')
```

Χωρίς KDE.
Κατακόρυφος άξονας (y) **λογαριθμικός.**, δηλαδή οι μικρές συχνότητες(π.χ. πολύ σπάνιες τιμές) μεγεθύνονται και οι πολύ μεγάλες μπάρες (π.χ. πλειοψηφικές τιμές) συμπιέζονται. Αυτό μας βοηθά στην καλύτερη κατανόηση της κατανομής.

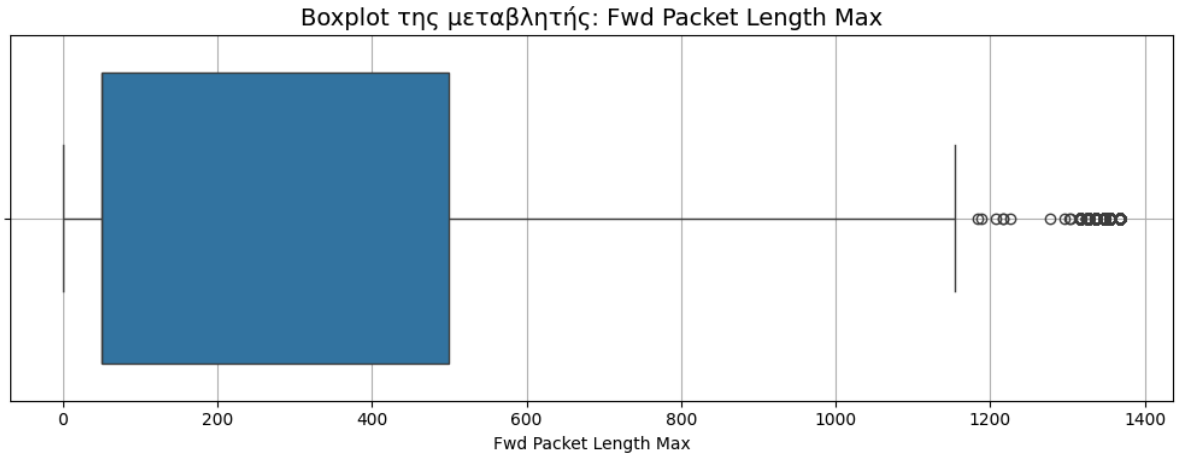


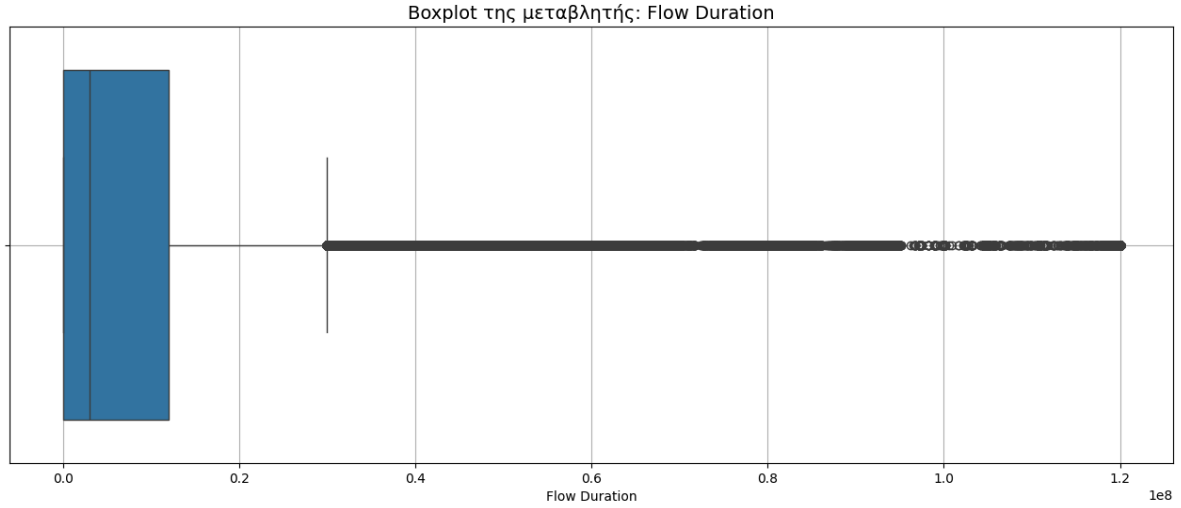
Boxplots

Τα **boxplots** χρησιμοποιήθηκαν για να αναδείξουμε τη διασπορά και τα άκρα των τιμών σε κάθε αριθμητική στήλη. Είναι ιδανικά για:

- εύκολη οπτική σύγκριση διάμεσου, τεταρτημορίων και outliers
- εντοπισμό μεταβλητών με μεγάλη ή ακραία διακύμανση

Τα boxplots χρησιμοποιήθηκαν επίσης για αριθμητικές μεταβλητές με επαρκή διασπορά (std > 0), ανεξαρτήτως του αριθμού μοναδικών τιμών. Είναι χρήσιμα για την ανάδειξη του εύρους τιμών, των τεταρτημορίων και των outliers, καθώς και για συγκρίσεις μεταξύ κλάσεων.





Countplots

Τα **countplots** εφαρμόστηκαν σε κατηγορικές στήλες όπως Label, Traffic Type, Dst Port κ.ά., για να δούμε:

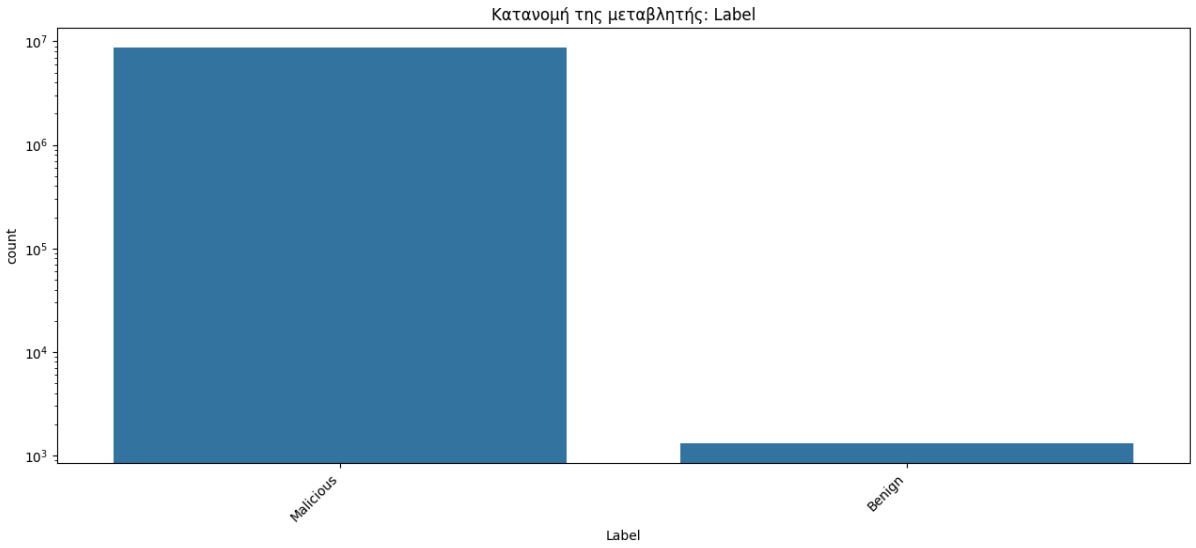
- πόσες φορές εμφανίζεται κάθε τιμή
- αν υπάρχει ανισορροπία κατηγοριών (class imbalance)
- ποιες κατηγορίες είναι οι πιο συχνές και ποιες σπάνιες

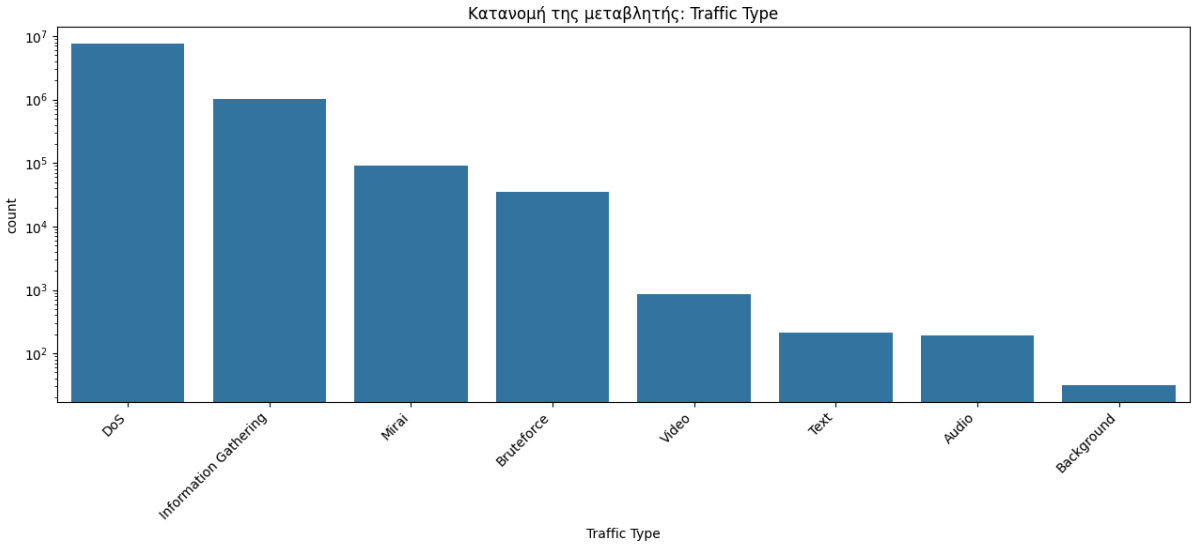
Τα countplots προτιμήθηκαν για κατηγορικές στήλες, ειδικά όταν ο αριθμός των μοναδικών τιμών ήταν μικρός. Επιλεκτικά παρουσιάζουμε παρακάτω την κατανομή των **Label** και **Traffic Type** που θα μας απασχολήσουν για την εκπαίδευση. Από τα countplots καταλαβαίνουμε ότι υπάρχει **ανισορροπία** δεδομένων-κατηγοριών ειδικά στην στήλη Label. Για την καλύτερη αναπαράσταση έχει χρησιμοποιηθεί λογαριθμική κλίμακα.

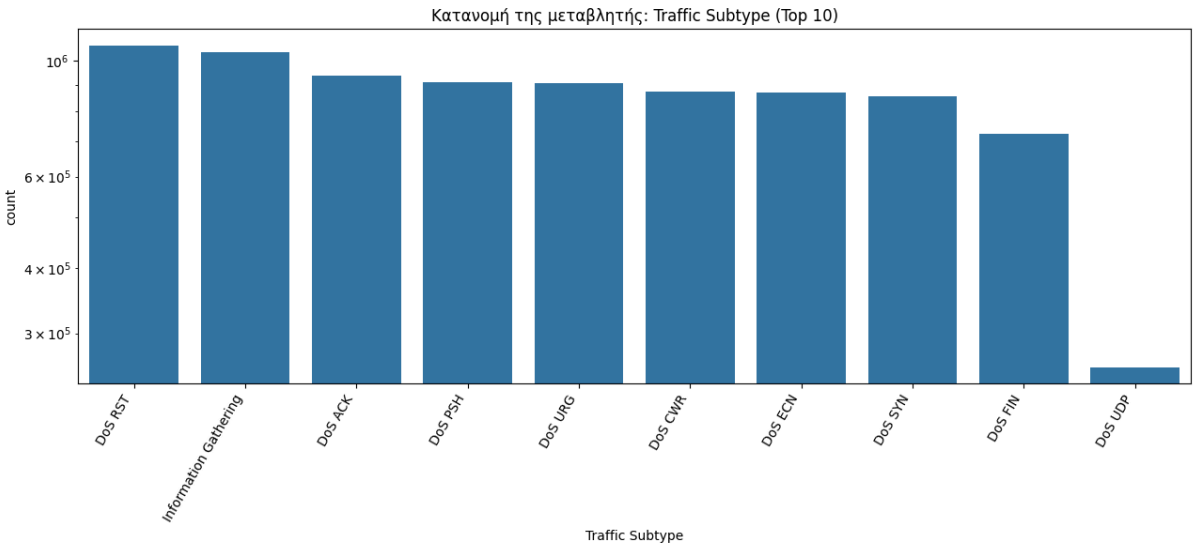
Με αυτο τον έλεγχο αν η στήλη έχει **πάνω από 20** διαφορετικές τιμές κρατάμε τις **Top 10** ,όπως φαίνεται και στο countplot του **Traffic Subtype** (εικόνα 3).

```
for col in categorical_cols:
    value_counts = df[col].value_counts()
    unique_vals = len(value_counts)

    # Αν έχει πολλές τιμές, κράτα μόνο τις 10 πιο συχνές
    if unique_vals > 20:
        top_vals = value_counts.nlargest(10).index
        plot_data = df[df[col].isin(top_vals)]
        title_note = " (Top 10)"
    else:
        plot_data = df
        title_note = ""
```







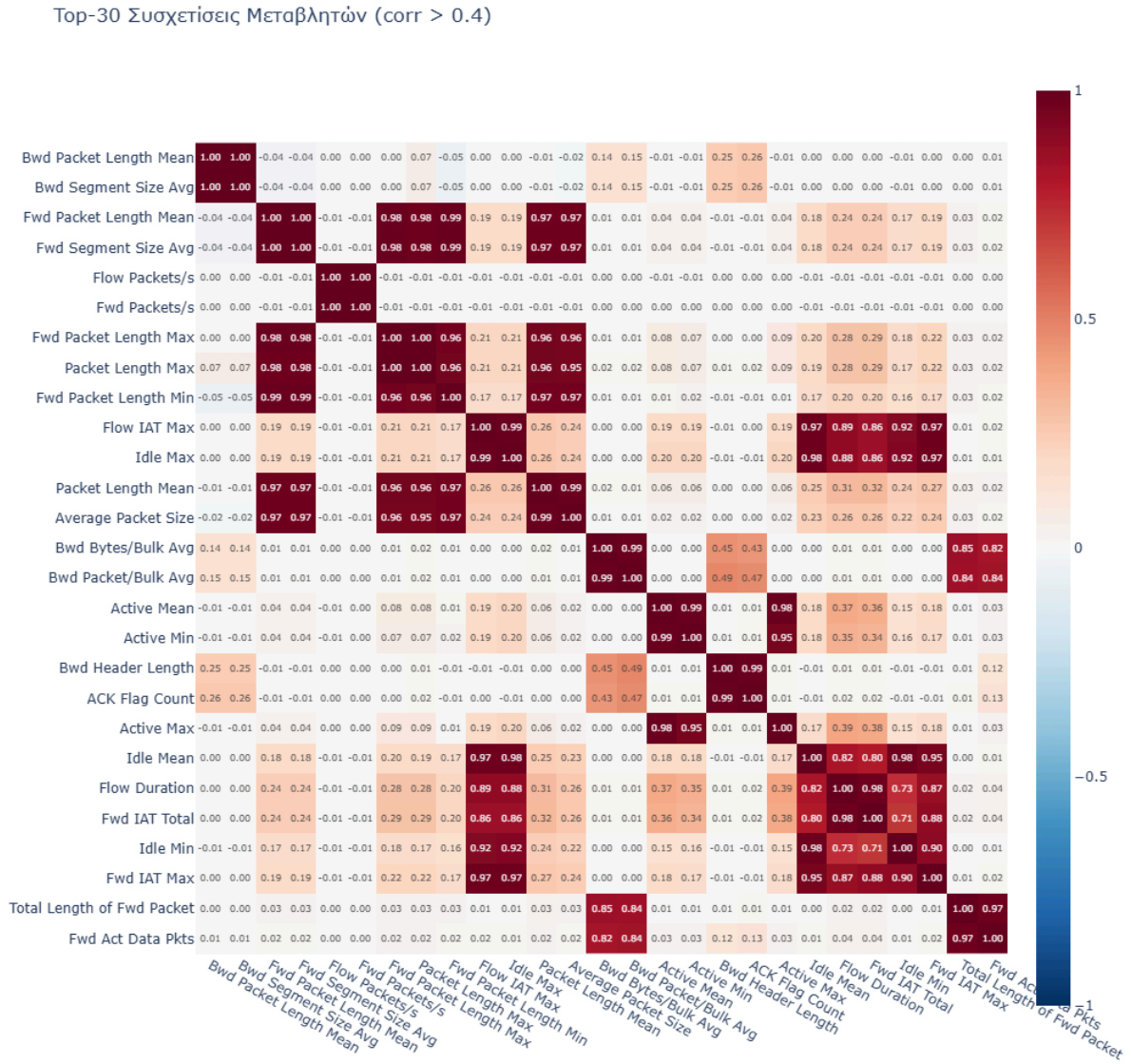
Heatmap

Το **heatmap** με βάση τον πίνακα συσχετίσεων (**correlation matrix**) μας επέτρεψε να:

- εντοπίσουμε πολύ συσχετισμένες στήλες
- εντοπίσουμε και να αφαιρέσουμε πλεονάζουσες μεταβλητές
- απλοποιήσουμε το dataset μέσω μείωσης διαστάσεων

Στην συνέχεια θα χρησιμοποιήσουμε τα **scores** απο τις συσχετίσεις (αυτές με score >0.95) για να μειώσουμε την διαστατικότητα.
Μαζί με το heatmap παράγουμε και το αρχείο *correlation_pairs_over_0.4_2025-05-18_00-24-21.txt*, το οποίο περιέχει κατά **φθίνουσα σειρά** τις συσχετίσεις μεταξύ των στηλών που είναι **άνω** του 0.4.

Ενδεικτικά :
Bwd Packet Length Mean→Bwd Segment Size Avg (1.00)
Fwd Packet Length Mean→Packet Length Mean(0.97)–Average Packet Size(0.97)–Fwd Packet Length Min(0.99)–Packet Length Max(0.98)–Fwd Packet Length Max(0.98)–Fwd Segment Size Avg(1.00)



Violin plots

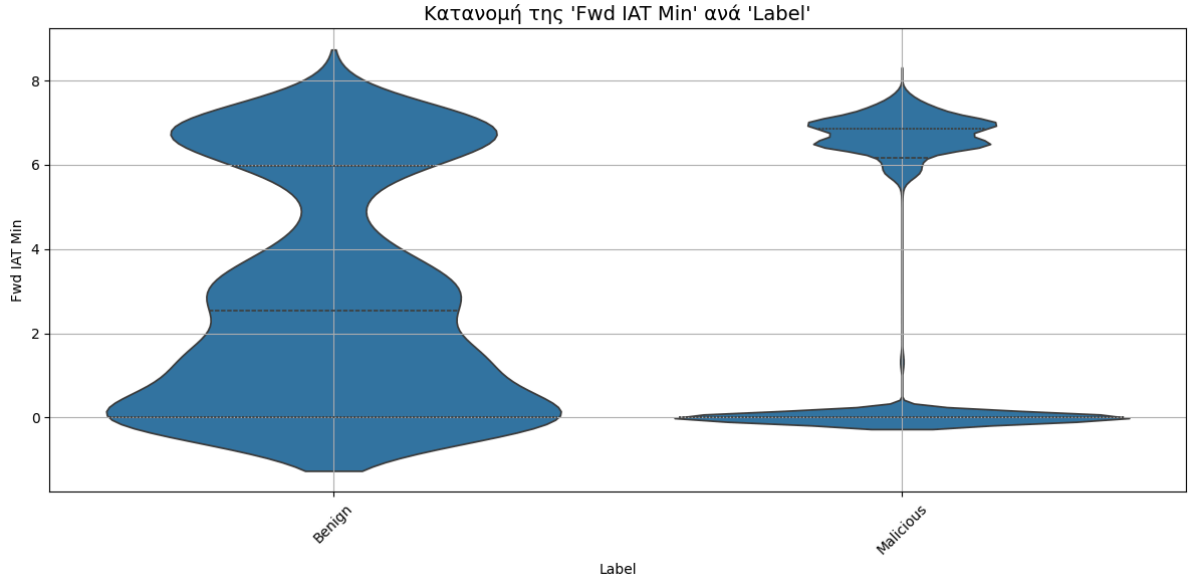
Τα **violin plots** προσφέρουν έναν συνδυασμό **boxplot + πυκνότητα κατανομής**, δίνοντάς μας πληροφορίες για:

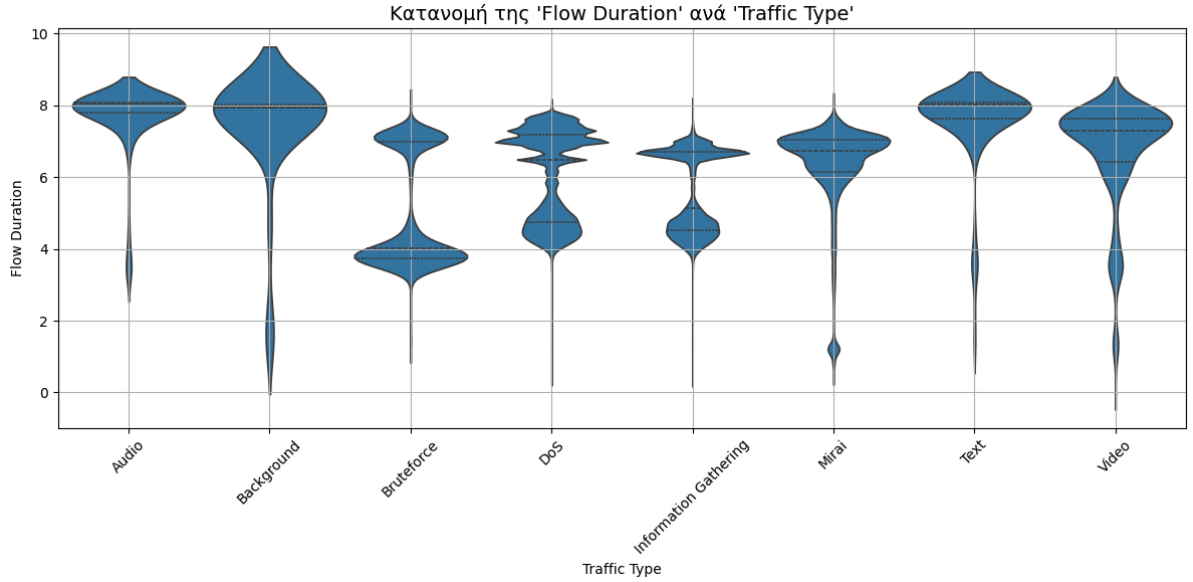
- τη μορφή της κατανομής (αν είναι ομοιόμορφη, διμορφική κ.λπ.)
- σύγκριση ανά κατηγορία, όπως Benign/Malicious ή Traffic Types
- πιο πλούσια οπτική πληροφορία από το απλό boxplot

Τα violin plots συνδύασαν αριθμητικά και κατηγορικά δεδομένα. Χρησιμοποιήθηκαν για αριθμητικές στήλες με διασπορά (std > 0), κατανεμημένες ανά τιμές κατηγορικών μεταβλητών **Label και Traffic type**, που μας ενδιαφέρουν για τα επόμενα ερωτήματα.

Για τον άξονα X χρησιμοποιούμε τις παρακάτω κατηγορικές στήλες.

```
categorical_targets = ['Label', 'Traffic Type', 'Traffic Subtype']
```





Ερώτημα 2

Μείωση διαστατικότητας

Αρχείο κώδικα: dimentionality.py
Αρχείο εισόδου: data.csv
Παραγόμενο αρχείο: pca_transformed_data_with_labels.csv, dataset_reduced_initial.csv, pca_explained_variance_plot.png

Για την μείωση της διαστατικότητας επιλέξαμε **2 τρόπους.**

Ο πρώτος είναι η διαγραφή στηλών χειροκίνητα (με κώδικα– **columns_to_drop_initial**), οι οποίες έχουν συσχέτιση **>=0.95** , βάσει του **correlation heatmap** που παράξαμε στο προηγούμενο ερώτημα ή δεν έχουν νόημα στην ανάλυση που θα πραγματοποιηθεί παρακάτω. Τελικό αρχείο: *dataset_reduced_initial.csv*

Πεδία ανούσια για ανάλυση–Αφαίρεση

– Αφαίρεση Κατηγορικών Πεδίων
Αφαιρούνται καθώς δεν περιέχουν χρήσιμη πληροφορία για ανάλυση:
Flow ID, Src IP, Dst IP, Timestamp, Traffic Subtype

– Αφαίρεση Flags
Αφαιρούνται καθώς δεν περιέχουν κάποια χρήσιμη πληροφορία και έχουν mean=0, std=0, max=0
Bwd PSH Flags, Fwd URG Flags,Bwd URG Flags, URG Flag Count, CWR Flag Count, ECE Flag Count

– Αφαίρεση Στηλών με Μηδενική Διακύμανση
Από τα περιγραφικά στατιστικά βλέπουμε ότι έχουν mean = std = 0 :
Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, Fwd Bulk Rate Avg κ.α

Επιπλέον, εφαρμόσαμε PCA στο **ήδη** μειωμένο αρχείο από την πρώτη τεχνική κρατώντας το 95% **(n_components=0.95)** της πληροφορίας.

```
pca = PCA(n_components=0.95)
X_pca_transformed = pca.fit_transform(X_scaled)
```

Το **PCA (Principal Component Analysis)** είναι μη εποπτευόμενη τεχνική μείωσης διαστάσεων, που προσπαθεί να βρει γραμμικούς συνδυασμούς των χαρακτηριστικών ώστε να εξηγήσει τη μέγιστη διασπορά των δεδομένων, μετά το PCA τα χαρακτηριστικά διατηρούν μόνο τη στατιστική πληροφορία. **Οι ετικέτες** είναι κατηγορικές και δεν έχουν γεωμετρική σημασία στον χώρο των χαρακτηριστικών. Όταν εφαρμόζουμε PCA, δημιουργούνται **νέες στήλες** (οι κύριες συνιστώσες) που είναι **γραμμικοί συνδυασμοί των αρχικών.**

Έτσι, **χωρίζουμε** τα χαρακτηριστικά (X) από τις ετικέτες ***y_label*** και ***y_traffic_type***.

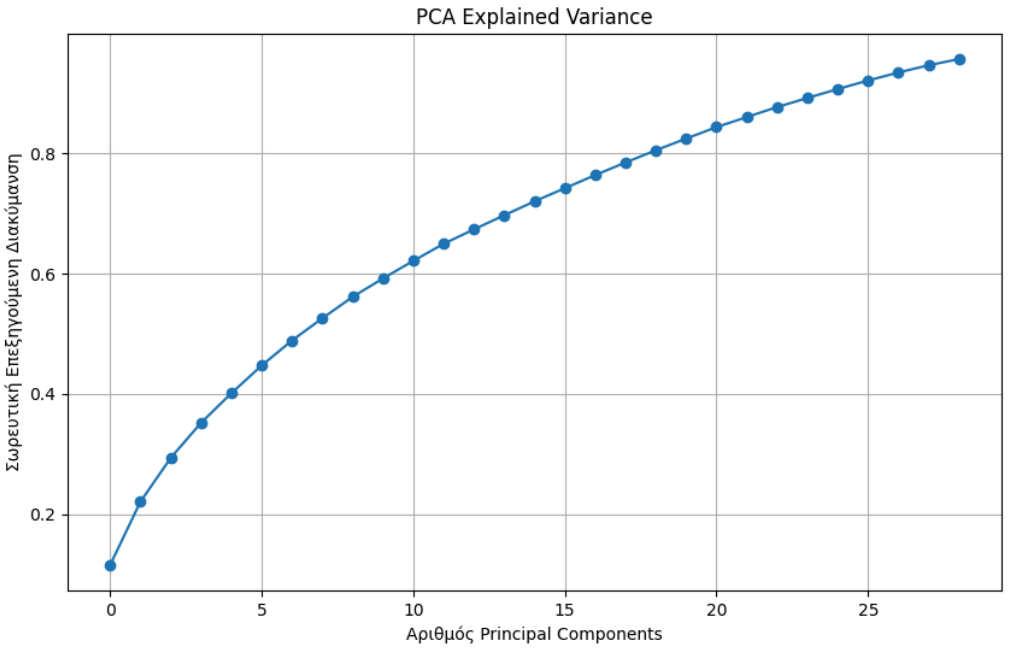
```
X = reduced_data.drop(columns=['Label', 'Traffic Type'], errors='ignore')
y_label = reduced_data['Label']
y_traffic_type = reduced_data['Traffic Type']
```

Έπειτα, με την χρήση του **StandardScaler** προχωράμε στην **κανονικοποίηση** για να μετασχηματιστούν τα χαρακτηριστικά ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1 (απαραίτητη διαδικασία γιατί το PCA ευαισθητοποιείται στις κλίμακες των χαρακτηριστικών-χαρακτηριστικά με μεγαλύτερη κλίμακα θα κυριαρχούσαν).

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns, index=X.index)
print("\nΤα χαρακτηριστικά κανονικοποιήθηκαν.")
```

Τέλος, δημιουργείται νέο DataFrame με τις κύριες συνιστώσες (PC1, PC2, ...P29) και προστίθενται οι ετικέτες **Label** και **Traffic Type** που είχαν αφαιρεθεί για να γίνει η διαδικασία του PCA.
Έχουμε προσθέσει και οπτικοποίηση, δημιουργώντας ένα **γράφημα** το οποίο δείχνει πόσο πληροφορία "κρατάμε" όσο αυξάνουμε τον αριθμό των PCA components.

Γράφημα PCA



–pca_explained_variance_plot

Το γράφημα ανεβαίνει **σταδιακά**, όπως πρέπει. Η καμπύλη αυτή δείχνει ότι με κάθε **νέο component** εξηγούμε και **περισσότερη πληροφορία**, αλλά με **φθίνουσα απόδοση**.
Component 10 – εξηγούμε περίπου **60%**.

Component 20 – εξηγούμε σχεδόν το **80–85%**.
Component 29 – εξηγούμε **πάνω από 90%**.

Βιβλιοθήκες

[Αναφορά σε όσες δεν έχουν ξανά αναφερθεί]

Στο αρχείο που εκτελούνται όλες οι παραπάνω διαδικασίες *dimentionality.py* εκτός απο βιβλιοθήκες που αναφέρθηκαν χρησιμοποιούμε τις:

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

Ο StandardScaler και το PCA είναι **εργαλεία** που προέρχονται από τη βιβλιοθήκη **scikit-learn.** Ο πρώτος ανήκει στο υποπακέτο **preprocessing**, ενώ ο δεύτερος στο **decomposition.** Ο **StandardScaler** κανονικοποιεί τα αριθμητικά χαρακτηριστικά, δηλαδή μετατρέπει κάθε στήλη ώστε να έχει **μέσο όρο 0** και **τυπική απόκλιση 1,** κάτι που είναι χρήσιμο για PCA (Principal Component Analysis). Το **PCA** το χρησιμοποιούμε ως τεχνική **μείωσης διαστάσεων** για να μετατρέψουμε τα δεδομένα σε λιγότερες "σύνθετες μεταβλητές" (components) που εξηγούν τη μεγαλύτερη δυνατή μεταβλητότητα.

Το αρχείο που παράγεται με την παραπάνω διαδικασία είναι το :
pca_transformed_data_with_labels.csv
το οποίο ορίζουμε σαν είσοδο στις παρακάτω διαδικασίες για μείωση των γραμμών πλέον.

Δειγματοληψία

Αρχείο κώδικα: sampling.py
Αρχείο εισόδου: pca_transformed_data_with_labels.csv
Παραγόμενο αρχείο: stratified_sample_combined.csv

Για τη δημιουργία ενός αντιπροσωπευτικού υποσυνόλου δεδομένων, εφαρμόστηκε διαδικασία δειγματοληψίας που συνδυάζει **στρωματοποιημένη** επιλογή, διατήρηση σπάνιων κατηγοριών και εξισορρόπηση. Αρχικά, φορτώθηκε το αρχικό σύνολο δεδομένων και δημιουργήθηκε η στήλη **Label_Type** με συνδυασμό των πεδίων Label και Traffic Type, ώστε να επιτραπεί η **λεπτομερέστερη στρωματοποίηση** και να διατηρηθούν οι **ισορροπίες** .

```
df["Label_Type"] = df["Label"] + " - " + df["Traffic Type"]
```

Στη συνέχεια, εντοπίστηκαν και διατηρήθηκαν όλες οι κατηγορίες που εμφανίζονταν **λιγότερες από 1000 φορές**, με στόχο τη **διατήρηση της πληροφορίας** σπάνιων αλλά σημαντικών περιπτώσεων.

```
type_counts = df["Label_Type"].value_counts()
rare_classes = type_counts[type_counts < 1000].index.tolist()
rare_df = df[df["Label_Type"].isin(rare_classes)]
main_df = df[~df["Label_Type"].isin(rare_classes)]
```

Από τις υπόλοιπες, λήφθηκε τυχαίο δείγμα 10% με διατήρηση της **αρχικής κατανομής** (stratify), ενώ οι σπάνιες κατηγορίες προστέθηκαν εξ ολοκλήρου στο σύνολο.

```
sample_frac = 0.10
sampled_df, _ = train_test_split(
    main_df,
    test_size=(1 - sample_frac),
    stratify=main_df["Label_Type"],
    random_state=42,
)
```

Ακολούθως, για την αντιμετώπιση της σημαντικής **ανισορροπίας** μεταξύ Malicious και Benign, πραγματοποιήθηκε **υποδειγματοληψία** του Malicious ώστε να διατηρηθεί μια προκαθορισμένη αναλογία, χωρίς όμως να αφαιρεθεί η ετερογένεια στους Traffic Types.

```
 === 6. Υποδειγματοληψία στο Malicious ===
malicious = df_final[df_final["Label"] == "Malicious"]
benign = df_final[df_final["Label"] == "Benign"]

desired_malicious = min(len(malicious), len(benign) * 7) #αναλογία
malicious_sampled = malicious.sample(n=desired_malicious, random_state=42)

df_final_balanced = pd.concat([benign, malicious_sampled],
ignore_index=True)
```

Η παραπάνω προσέγγιση εξασφαλίζει την επάρκεια πληροφορίας, τη μείωση όγκου και τη βελτίωση της μάθησης μέσω περιορισμού των συχνά κυρίαρχων κατηγοριών. Επιλέξαμε την **υποδειγματοληψία** σε αυτή την περίπτωση και δεν προσθέσαμε τεχνητά δείγματα στις σπάνια εμφανιζόμενες κατηγορίες, ώστε να διατηρήσουμε τη **φυσική κατανομή** των δεδομένων αυτούσια και χωρις να χάνουμε πληροφορία ακόμη και αν δεν εμφανίζεται τόσο συχνά.

Με την βοήθεια του αρχείου *test_counts.py* βλέπουμε την κατανομή στο τελικό αρχείο. Το τελικό δείγμα μεγέθους ~10.000 εγγραφών επιλέχθηκε ως πιο κατάλληλο μετά από πολλές παραδοχές για εκπαίδευση, καθώς **διατηρεί όλα τα Benign** δείγματα και **διασφαλίζει ποικιλία επιθέσεων χωρίς να λείπει καμία κατηγορία** (Traffic Type), ενώ περιορίζει το Malicious με **αναλογία ~7:1**. Αυτό επιτρέπει **γρήγορη** εκπαίδευση και αξιόπιστη αξιολόγηση σε ταξινομητές όπως **SVM και NN,** χωρίς ανάγκη για πρόσθετες τεχνικές εξισορρόπησης.

Κατανομές

```
Κατανομή Label:
Label
Malicious    9107
Benign       1301
Name: count, dtype: int64

Κατανομή Traffic Type (top 10):
Traffic Type
DoS                      7888
Information Gathering    1083
Video                     870
Text                      209
Audio                     190
Mirai                      99
Bruteforce                 37
Background                 32
Name: count, dtype: int64

Σύνολο γραμμών: 10,408
```

Βιβλιοθήκες

[Αναφορά σε όσες δεν έχουν ξανά αναφερθεί]

```
from sklearn.model_selection import train_test_split
```

Για τη δειγματοληψία επιλέχθηκε η χρήση της **train_test_split** από τη βιβλιοθήκη **scikit-learn**, με στρωματοποίηση ως προς τη σύνθετη κατηγορία **Label_Type**, ώστε να διατηρηθούν οι αναλογίες μεταξύ των τύπων κυκλοφορίας, όπως προαναφέρθηκε.

MiniBatchKMeans

Αρχείο κώδικα: clustering_miniBatchKmeans.py
Αρχείο εισόδου: pca_transformed_data_with_labels.csv
Παραγόμενο αρχείο: minibatch_kmeans_balanced_with_smote.csv

Για τη μείωση του μεγέθους του dataset χρησιμοποιήθηκε ο **MiniBatchKMeans** αντί του απλού **KMeans**, λόγω της υψηλής αποδοτικότητάς του σε datasets μεγάλης κλίμακας. Ενώ πρόκειται για προσεγγιστικό αλγόριθμο, η διαφορά στην ακρίβεια είναι αμελητέα, ενώ τα οφέλη σε **χρόνο** και **μνήμη** είναι καθοριστικά για το μέγεθος των αρχικών δεδομένων. Ο **MiniBatchKMeans** δουλεύει με **μικρά τυχαία δείγματα** των δεδομένων (mini-batches) σε κάθε επανάληψη, μειώνοντας έτσι σημαντικά τον χρόνο και τη μνήμη που απαιτείται για την ομαδοποίηση. Αυτή η προσέγγιση είναι ιδανική για μεγάλα datasets, όπως το δικό μας, όπου η χρήση του απλού KMeans θα ήταν πολύ πιο αργή και απαιτητική σε πόρους. Με τον MiniBatchKMeans χωρίσαμε τα δεδομένα σε **10.000** ομάδες και από κάθε ομάδα διαλέξαμε τα 8 πιο αντιπροσωπευτικά δείγματα. Έτσι, θα διατηρηθεί όσο το δυνατόν περισσότερη ποικιλία από **traffic patterns** και θα επιτευχθεί καλύτερη ταξινόμηση στα μοντέλα που θα εκπαιδευτούν με αυτά τα δεδομένα, έχοντας και καλή απόδοση σε χρόνο - μνήμη.

Εφαρμόσαμε επίσης τεχνικές για να αντιμετωπίσουμε την ανισορροπία των δεδομένων όπως έχουν προκύψει από τα διαγράμματα (countplots, histograms) ώστε πέρα από την συσταδοποίηση που κάναμε να είναι “έτοιμα” τα δεδομένα μας για εκπαίδευση. Συγκεκριμένα, εφαρμόστηκε **SMOTE** για τεχνητή ενίσχυση. Η αύξηση μέσω SMOTE γεννάει νέα παραδείγματα με interpolation, που τα χρειαζόμαστε για να φτιάξουμε ένα dataset αντιπροσωπευτικό με αρκετες γραμμές για εκπαίδευση.

Επιλέξαμε να αυξήσουμε τις στήλες με λιγότερες εμφανίσεις και όχι να μειώσουμε την εμφάνιση σε αυτές με τις πολλές. Για σωστή εκπαίδευση θελουμε πάνω από το 10.000 γραμμές, οπότε αν μειώναμε μόνο τις στήλες με τις πολλές εμφανίσεις δεν θα ισορροπούσε το αρχείο (Background–32 εμφανίσεις, Dos–7 εκατομμύρια εμφανίσεις) και θα ήταν σίγουρα κάτω από το θεμιτό όριο γραμμών.

Ρυθμίσεις που έχουμε επιλέξει:

Μέγιστος αριθμός clusters ανά τύπο κίνησης.

```
max_clusters_per_type = 300
```

Αριθμός clusters για Benign traffic

```
benign_clusters = 1500
```

Κατώφλι για clustering.Αν ένα traffic type έχει τουλάχιστον 2000 δείγματα, τότε εφαρμόζεται clustering.Αν έχει λιγότερα, θεωρείται "σπάνιο" και ενισχύεται μέσω SMOTE.

```
min_points_threshold = 2000
```

Αριθμός δειγμάτων που επιλέγονται από κάθε cluster για να αποφευχθεί υπερβολική ποσότητα δεδομένων. Επιλέγονται μόνο τα 8 πλησιέστερα σημεία στο centroid κάθε cluster

```
num_points_per_cluster = 8
```

Τιμή για επαναληψιμότητα. Χρησιμοποιείται για να εξασφαλιστεί ότι τα αποτελέσματα είναι επανεκτελέσιμα.

```
random_state = 42
```

Μέγεθος παρτίδας για το MiniBatchKMeans.Το MiniBatchKMeans επεξεργάζεται τα δεδομένα σε "παρτίδες" (mini-batches) για βελτίωση ταχύτητας και μείωση μνήμης. Εδώ, κάθε παρτίδα έχει έως και 10.000 σημεία.

```
batch_size = 10000
```

Για το **Benign** με συνολικό αριθμό δειγμάτων **=1.302<<Malicious** εφαρμόστηκε MiniBatchKMeans clustering με αριθμό clusters ίσο με τον αριθμό των δειγμάτων (1302), για συνέπεια με τη διαδικασία που ακολουθήθηκε στους Malicious τύπους και από κάθε cluster επιλέχθηκε τα 8 κοντινότερα σημεία στο κέντρο του (αλλά στην πράξη χρησιμοποιήθηκαν όλα τα διαθέσιμα σημεία–1 από κάθε cluster).

```
# === 3. Clustering σε Benign ===
print(f"\nBenign → {benign_clusters} clusters")
X = benign_df[pca_cols].values
kmeans = MiniBatchKMeans(n_clusters=min(benign_clusters, len(X)),
                         batch_size=batch_size,
random_state=random_state)
labels = kmeans.fit_predict(X)
benign_df['Cluster'] = labels
centroids = kmeans.cluster_centers_

for i in tqdm(range(kmeans.n_clusters), desc="Benign - closest points"):
    group = benign_df[benign_df['Cluster'] == i]
    if group.empty: continue
    dists = cdist([centroids[i]], group[pca_cols])
    closest_idxs = np.argsort(dists[0])[:num_points_per_cluster]
    for idx in closest_idxs:
        selected_rows.append(group.iloc[idx])
```

Για το **Malicious** χωρίσαμε τα δεδομένα ανά Traffic Type

– Για traffic types με **< 2000 δείγματα**, εφαρμόστηκε **SMOTE** για τεχνητή ενίσχυση, και επιλέχθηκαν έως 100 συνθετικά δείγματα ανά κατηγορία.Η αύξηση μέσω SMOTE γεννάει νέα παραδείγματα με **interpolation**, δεν κάνει απλό duplication.

– Για traffic types με **≥ 2000 δείγματα**, εφαρμόστηκε MiniBatchKMeans clustering (έως **300 clusters ανά κατηγορία**) και από κάθε cluster επιλέχθηκαν τα **8 πλησιέστερα σημεία** προς το κέντρο του cluster, ώστε να διασφαλιστεί αντιπροσωπευτικότητα.

```
# === 4. Clustering ή SMOTE ανά Traffic Type ===
print(f"\nMalicious ανά Traffic Type")

for traffic_type, group in malicious_df.groupby('Traffic Type'):
    print(f"Επεξεργασία: {traffic_type} ({len(group)} σημεία)")

    if len(group) < min_points_threshold:
        print(f"{traffic_type} → κάτω από threshold
({min_points_threshold}) → SMOTE")

        try:
            le = LabelEncoder()
            group["Traffic_Type_Label"] =
le.fit_transform([traffic_type] * len(group))
            smote = SMOTE(random_state=random_state)
            X_resampled, y_resampled =
smote.fit_resample(group[pca_cols], group["Traffic_Type_Label"])
            group_resampled = pd.DataFrame(X_resampled,
columns=pca_cols)
            group_resampled['Label'] = 'Malicious'
            group_resampled['Traffic Type'] = traffic_type

            # Κρατάμε μέχρι 100 από το SMOTE output

selected_rows.extend(group_resampled.sample(n=min(len(group_resampled),
100), random_state=42).to_dict(orient='records'))

        except Exception as e:
            print(f"SMOTE απέτυχε για {traffic_type}: {e}")
            print(f"Κρατάμε {len(group)} αρχικά σημεία ως fallback")
            selected_rows.extend(group.to_dict(orient='records'))

        continue

    # Αν έχει αρκετά σημεία → Clustering
    print(f"Clustering σε {max_clusters_per_type} clusters")
    X = group[pca_cols].values
    kmeans = MiniBatchKMeans(n_clusters=min(max_clusters_per_type,
len(X)),
                             batch_size=batch_size,
random_state=random_state)
```

```
    labels = kmeans.fit_predict(X)
    group['Cluster'] = labels
    centroids = kmeans.cluster_centers_

    for i in range(kmeans.n_clusters):
        points = group[group['Cluster'] == i]
        if points.empty: continue
        dists = cdist([centroids[i]], points[pca_cols])
        closest_idxs = np.argsort(dists[0])[:num_points_per_cluster]
        for idx in closest_idxs:
            selected_rows.append(points.iloc[idx])
```

Η τελική κατανομή των δεδομένων, προερχόμενη από **clustering** και **επιλεκτική εφαρμογή SMOTE**, οδηγεί σε ένα dataset με σαφώς πιο ισορροπημένες κλάσεις. Η κατηγορία Malicious έχει πλέον ικανοποιητική αναλογία ως προς τα Benign, ενώ οι κυριότερες επιθέσεις (DoS, Information Gathering, Mirai, Bruteforce) έχουν περίπου ίση κατανομή (~2400 δείγματα). Αυτό ενισχύει την ικανότητα του μοντέλου να διακρίνει μεταξύ των βασικών τύπων επιθέσεων, ενώ η περιορισμένη παρουσία λιγότερο κοινών κατηγοριών (π.χ. Audio, Text) συμβάλλει σε καλύτερη γενίκευση, χωρίς να χάνουμε καμία κατηγορία.

Κατανομές

```
Κατανομή Label:
Label
Malicious    9586
Benign       1301
Name: count, dtype: int64

Κατανομή Traffic Type (top 10):
Traffic Type
Information Gathering    2400
Mirai                    2400
DoS                      2400
Bruteforce               2386
Video                     870
Text                      209
Audio                     190
Background                 32
Name: count, dtype: int64

Σύνολο γραμμών: 10,887
```

Βιβλιοθήκες
[Αναφορά σε όσες δεν έχουν ξανά αναφερθεί]

```
from sklearn.cluster import MiniBatchKMeans
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from scipy.spatial.distance import cdist
from tqdm import tqdm
```

Η βιβλιοθήκη **scikit-learn** είναι για προεπεξεργασία δεδομένων. Εμείς χρησιμοποιούμε το **MiniBatchKMeans (sklearn.cluster)** για clustering μεγάλων συνόλων δεδομένων και το **LabelEncoder (sklearn.preprocessing)** , ο οποίος είναι ένας απλός encoder που μετατρέπει τις κατηγορικές ετικέτες (π.χ. "DoS") σε αριθμούς. Απαιτείται από τον SMOTE που χρησιμοποιούμε, ο οποίος λειτουργεί μόνο με αριθμητικές ετικέτες και χρησιμοποιείται μόνο τοπικά.

Η βιβλιοθήκη **imbalanced-learn** είναι εξειδικευμένη βιβλιοθήκη για την αντιμετώπιση ανισόρροπων datasets. Εμείς χρησιμοποιούμε το **SMOTE (imblearn.over_sampling),** το οποίο εφαρμόζεται για δημιουργία τεχνητών (συνθετικών) δειγμάτων στην υποεκπροσωπούμενη κατηγορία (π.χ Benign), με σκοπό την εξισορρόπηση του dataset.

Η βιβλιοθήκη **SciPy** παρέχει εργαλεία για αριθμητικούς υπολογισμούς. Εμείς χρησιμοποιούμε το **cdist (scipy.spatial.distance),** το οποίο υπολογίζει αποστάσεις) μεταξύ δύο συνόλων σημείων. Την χρειαζόμαστε γιατί για κάθε cluster υπολογίζεται η απόσταση κάθε σημείου από το κέντρο.

Η βιβλιοθήκη **tqdm** προσφέρει ένα **progress bar** για την παρακολούθηση της προόδου σε loops, κάτι ιδιαίτερα χρήσιμο για εμάς αφού η επεξεργασία δεδομένων διαρκεί πολύ.

HDBSCAN

Αρχείο κώδικα: clustering_hdbscan.py
Αρχείο εισόδου: pca_transformed_data_with_labels.csv
Παραγόμενο αρχείο: hdbscan_chunks_output\final_hdbscan_balanced.csv

Για την περαιτέρω επεξεργασία και συμπίεση του συνόλου δεδομένων που προέκυψε από την μείωση διαστατικότητας, εφαρμόστηκε μια στρατηγική βασιζόμενη στον αλγόριθμο clustering **HDBSCAN,** ενώ ενσωματώθηκαν και **τεχνικές εξισορρόπησης** για να φτιάξουμε αντιπροσωπευτικά αρχεία για εκπαίδευση κατηγοριοποιητών.Η επιλογή του **HDBSCAN** έγινε λόγω της ικανότητάς του να εντοπίζει clusters διαφορετικής πυκνότητας και να εξαιρεί θόρυβο (noise points), χωρίς την ανάγκη επιλογής του ενοχλητικού **hyperparameter ε** (epsilon), όπως στο **DBSCAN.** Επιπλέον, είναι πιο **σταθερός** σε μεγάλα datasets λόγω της ιεραρχικής προσέγγισης του και πιο **γρήγορος**. Το αρχείο χωρίστηκε σε 9 τμήματα (chunks) των 1.000.000 γραμμών για αποδοτική επεξεργασία μνήμης και εξοικονόμηση χρόνου. Σε κάθε **chunk** εφαρμόστηκε ο αλγόριθμος HDBSCAN (με min_cluster_size=50), ο οποίος εντοπίζει δυναμικά περιοχές **υψηλής πυκνότητας** χωρίς να απαιτεί προκαθορισμένο αριθμό clusters.

```
chunk_size = 1_000_000
num_chunks = 9
min_cluster_size = 50
points_per_cluster = 5
rare_threshold = 1500
random_state = 42
```

Από κάθε cluster επιλέχθηκαν τα **5 σημεία** πλησιέστερα στο κέντρο (centroid), ώστε να διατηρηθούν οι πιο "αντιπροσωπευτικές" εγγραφές.

```
def find_top_n_points_per_cluster(df, pca_cols, n=5):
    selected = []
    for _, group in df.groupby("Cluster"):
        X = group[pca_cols].to_numpy()
        centroid = np.mean(X, axis=0)
        group = group.copy()
        group["dist"] = np.linalg.norm(X - centroid, axis=1)
        top_n = group.nsmallest(n, "dist")
        selected.append(top_n.drop(columns=["dist"]))
    return pd.concat(selected)
```

Παράλληλα, εντοπίστηκαν και διατηρήθηκαν όλες οι εγγραφές που ανήκουν σε **σπάνιους τύπους** κυκλοφορίας (Traffic Type < 1500 εγγραφές), ανεξαρτήτως αν εντοπίστηκαν από τον HDBSCAN, με στόχο **να μη χαθεί πολύτιμη πληροφορία** από υποεκπροσωπούμενες κλάσεις.

```
# === Εντοπισμός και αποθήκευση σπάνιων Traffic Types ===
rare_traffic_types = df_all["Traffic Type"].value_counts()
rare_traffic_types = rare_traffic_types[rare_traffic_types <
rare_threshold].index.tolist()
```

```
rare_rows = df_all[df_all["Traffic Type"].isin(rare_traffic_types)].copy()
```

Αφού έχουμε εκτελέσει τον **HDBSCAN** προχωράμε να αντιμετωπίσουμε την ανισορροπία των δεδομένων.

Εφαρμόζουμε "ήπιο" (1302->2000) oversampling μόνο των μειοψηφικών Label – Benign με duplication. Η τελική αύξηση της κατηγορίας **Benign** προέκυψε αφενός από την εφαρμογή **oversampling έως 2000 εγγραφές** (όταν το αρχικό πλήθος είναι μικρότερο–εδώ 1302), και αφετέρου από την **ενσωμάτωση σπάνιων τύπων κυκλοφορίας** (rare Traffic Types) που φέρουν την ετικέτα Benign. Η ενίσχυση αυτή δεν είναι τεχνητή, καθώς τα σημεία προέρχονται από υπαρκτά δεδομένα του αρχικού συνόλου.

```
if len(df_minor) < 2000:
    df_minor_oversampled = df_minor.sample(n=2000, replace=True)
```

Αυτό το τμήμα του κώδικα εφαρμόζει μια **στοχευμένη τεχνική εξισορρόπησης** για κατηγορίες **Traffic Type** που έχουν μικρή εκπροσώπηση στο τελικό δείγμα (merged_df). Αρχικά υπολογίζεται πόσες εγγραφές έχει κάθε Traffic Type και εντοπίζονται εκείνοι με λιγότερες από 500 εμφανίσεις. Για κάθε μία από αυτές τις κατηγορίες, ελέγχεται αν υπάρχουν επιπλέον δείγματα στο αρχικό dataset (df_all). Αν ο συνολικός διαθέσιμος αριθμός είναι ≤500, τότε ορίζεται **στόχος τα 500 δείγματα**. Αν υπάρχει περισσότερη διαθεσιμότητα, επιδιώκεται ενίσχυση **έως και 1.000 εγγραφές**. Ο απαραίτητος αριθμός συμπληρώνεται με **duplication υπαρχόντων δειγμάτων** από το **merged_df**, με χρήση δειγματοληψίας με επανάθεση (replace=True). Έτσι, εξασφαλίζεται ότι ακόμη και οι λιγότερο συχνοί τύποι κυκλοφορίας θα έχουν **επαρκή παρουσία στο τελικό dataset**, χωρίς την ανάγκη για τεχνητή παραγωγή δεδομένων, ενισχύοντας τη δυνατότητα γενίκευσης των ταξινομητών σε σπάνιες κατηγορίες.

```
# Υπολογίζουμε πλήθος εμφανίσεων ανά traffic type στο merged_df
merged_counts = merged_df["Traffic Type"].value_counts()
under_500 = merged_counts[merged_counts < 500]

for traffic_type, current_len in under_500.items():
    # Πόσα έχει στο αρχικό df_all
    total_in_all = df_all[df_all["Traffic Type"] == traffic_type].shape[0]

    # Αν το αρχικό έχει <= 500 → στόχος είναι 500
    if total_in_all <= 500:
        target_n = 500
    else:
        # Αν έχει >500 → αύξησέ το μεχρι 1000
         target_n = min(1000, total_in_all)

    needed = target_n - current_len
```

```
    if needed <= 0:
        continue

    df_subset = merged_df[merged_df["Traffic Type"] == traffic_type]

    if df_subset.empty:
        print(f"Το '{traffic_type}' δεν υπάρχει καθόλου στο merged_df -
αγνοείται.")
        continue

    duplicated = df_subset.sample(n=needed, replace=True,
random_state=random_state)
    merged_df = pd.concat([merged_df, duplicated], ignore_index=True)
    print(f"{traffic_type}': {current_len} → {current_len + needed}
(στόχος: {target_n})")
```

Επιπλέον, περιορίστηκε η υπερβολική παρουσία της κατηγορίας Malicious-DoS κατά 20%, εξισορροπώντας την κατανομή μεταξύ επιθέσεων.

```
mask_malicious_dos = (merged_df["Label"] == "Malicious") &
(merged_df["Traffic Type"] == "DoS")
df_malicious_dos = merged_df[mask_malicious_dos]
keep_n = int(len(df_malicious_dos) * 0.8)

df_malicious_dos_reduced = df_malicious_dos.sample(n=keep_n,
random_state=random_state)
merged_df = pd.concat([merged_df[~mask_malicious_dos],
df_malicious_dos_reduced], ignore_index=True)
```

Η τελική κατανομή του συνόλου δεδομένων περιλαμβάνει 42.312 εγγραφές Malicious και 4.370 Benign, με συνολικό μέγεθος 46.682 γραμμών. Παρότι η αναλογία Malicious : Benign εξακολουθεί να είναι ασύμμετρη, το **πλήθος των Benign** είναι επαρκές για την εκπαίδευση ταξινομητών, ενώ η διατήρησή τους έγινε μέσω **ελεγχόμενου oversampling.** Οι κυρίαρχες κατηγορίες επιθέσεων (DoS, Information Gathering) διατηρούν μεγάλη αντιπροσώπευση, ενώ σπανιότεροι Traffic Types όπως Audio, Text και Background ενισχύθηκαν **επιλεκτικά με duplication** ώστε να έχουν τουλάχιστον **500 σημεία** η καθεμία. Η στρατηγική αυτή επιτρέπει στα μοντέλα να μαθαίνουν αποτελεσματικά τόσο από συχνές όσο και από σπάνιες κατηγορίες, **χωρίς υπερβολική τεχνητή αλλοίωση των δεδομένων.** Άρα, το τελικό σύνολο πετυχαίνει πληρότητα και επαρκή εκπροσώπηση όλων των σημαντικών κατηγοριών

Κατανομές

```
Κατανομή Label:
Label
Malicious    42312
Benign        4370
Name: count, dtype: int64

Κατανομή Traffic Type (top 10):
Traffic Type
DoS                      27088
Information Gathering    13224
Video                     2870
Bruteforce                1000
Mirai                     1000
Audio                      500
Background                 500
Text                       500
Name: count, dtype: int64

Σύνολο γραμμών: 46,682
```

Βιβλιοθήκες

[Αναφορά σε όσες δεν έχουν ξανά αναφερθεί]

```
from hdbscan import HDBSCAN
from sklearn.utils import shuffle
```

Η βιβλιοθήκη **hdbscan** χρησιμοποιήθηκε για την ομαδοποίηση των σημείων βάσει πυκνότητας, χωρίς να απαιτείται εκ των προτέρων αριθμός clusters, επιτρέποντας τον εντοπισμό αντιπροσωπευτικών περιοχών και σημείων σε πολύπλοκα υποσύνολα δεδομένων. Αντίστοιχα, η συνάρτηση **shuffle()** από το **scikit-learn** χρησιμοποιήθηκε για την τυχαία αναδιάταξη του τελικού συνόλου, αποτρέποντας **πιθανό bias** κατά τη φάση της εκπαίδευσης των ταξινομητών.

Ερώτημα 3

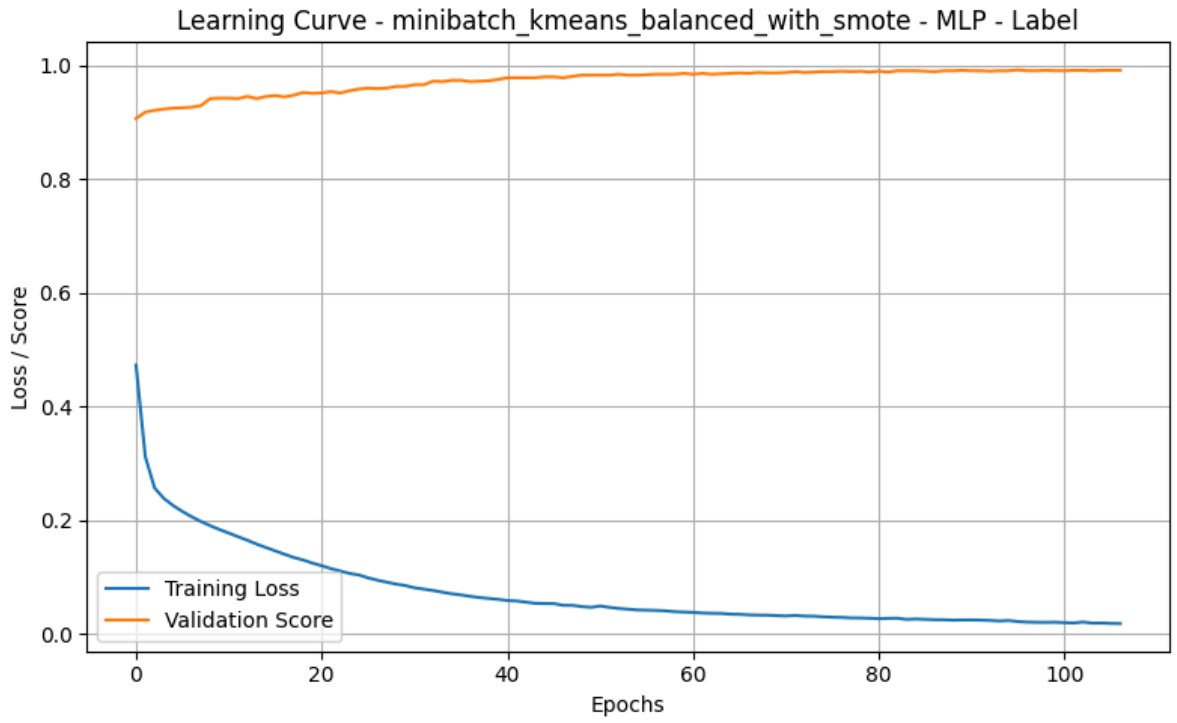
Εκπαίδευση κατηγοριοποιητή βασισμένο σε Neural Networks:

Χρησιμοποίησαμε κατηγοριοποιητή **MLPClassifier**, ο οποίος δομήθηκε με δύο κρυφά επίπεδα (μεγέθους 50 και 25 νευρώνες αντίστοιχα), χρησιμοποιώντας τη **συνάρτηση ενεργοποίησης** 'relu' και τον **βελτιστοποιητή** 'adam' για την προσαρμογή των βαρών. Προκειμένου να διασφαλιστεί η βέλτιστη γενίκευση και να αποφευχθεί η υπερεκπαίδευση, εφαρμόστηκε η **τεχνική της πρόωρης διακοπής** (early stopping). Η απόδοση του μοντέλου αξιολογήθηκε, χρησιμοποιώντας **classification reports** και **confusion matrices** για τα σύνολα **training**, **validation** και **testing**, παρέχοντας μια ολοκληρωμένη εικόνα της ικανότητάς του να ταξινομεί δεδομένα για '**Label**' και '**Traffic Type**'.

➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε ο MiniBatchKmeans

Καμπύλες μάθησης

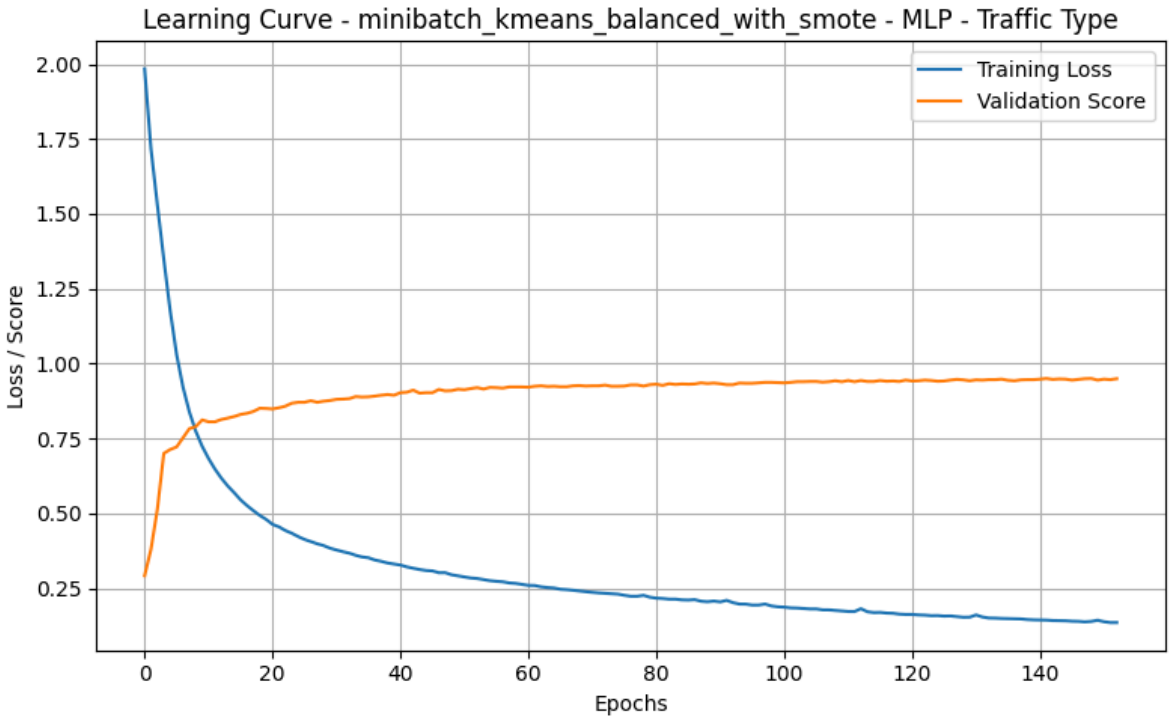
Label



Αυτό το διάγραμμα δείχνει πώς μαθαίνει το μοντέλο να ταξινομεί σωστά το "**Label**" καθώς περνούν οι εποχές εκπαίδευσης.
Το **Training Loss** (μπλε γραμμή) δείχνει ότι το σφάλμα του μοντέλου στα δεδομένα εκπαίδευσης μειώνεται γρήγορα και σταθερά, κάτι που σημαίνει ότι το μοντέλο "μαθαίνει καλά" από τα δεδομένα του.

Το **Validation Score** (πορτοκαλί γραμμή) δείχνει την απόδοσή του στα δεδομένα επικύρωσης, δηλαδή σε δεδομένα που δεν έχει δει πριν. Αυτή η γραμμή ανεβαίνει απότομα στην αρχή και φτάνει πολύ κοντά στο **1.0**, που σημαίνει σχεδόν τέλεια απόδοση. Επιπλέον, παραμένει **σταθερά υψηλή**, κάτι που δείχνει ότι το μοντέλο δουλεύει καλά όχι μόνο με τα δεδομένα που έχει μάθει, αλλά και με καινούρια.
Επειδή η **διαφορά** ανάμεσα στις δύο γραμμές είναι **μικρή** και η **απόδοση** παραμένει ψηλά, καταλαβαίνουμε ότι το μοντέλο είναι ισορροπημένο και αποδοτικό. Δεν κάνει **overfitting** τα εκπαιδευτικά δεδομένα, αλλά καταφέρνει να γενικεύει σωστά.

Traffic Type



Το παραπάνω διάγραμμα παρουσιάζει την πορεία εκπαίδευσης του ίδιου **MLP** μοντέλου, με στόχο αυτή τη φορά την πρόβλεψη της κατηγορίας "**Traffic Type**", η οποία περιλαμβάνει περισσότερες και πιο εξειδικευμένες τάξεις. Εδώ, το **training loss** ξεκινά από το 2.0, γεγονός που υποδεικνύει ότι αρχικά το μοντέλο δυσκολεύεται περισσότερο να μάθει από τα δεδομένα.
Με την πάροδο των **εποχών**, παρατηρείται σταθερή και συνεχιζόμενη μείωση της απώλειας εκπαίδευσης, η οποία πλησιάζει το **0.15**, χωρίς ωστόσο να πέφτει τόσο χαμηλά όσο στην περίπτωση του "**Label**". Η καμπύλη **validation score** ακολουθεί ανοδική πορεία και σταθεροποιείται γύρω στο **0.95**, δείχνοντας ότι το μοντέλο τελικά καταφέρνει να γενικεύσει ικανοποιητικά και σε πιο απαιτητικές κλάσεις.
Παρόλο που η μάθηση φαίνεται να είναι πιο δύσκολη, **δεν** εμφανίζεται **overfitting** – οι καμπύλες παραμένουν κοντά και σταθερές. Αυτό επιβεβαιώνει ότι οι επιλογές **early stopping** και **επαλήθευσης** (validation) ήταν αποτελεσματικές.

Μετρικές Ταξινόμησης

Το **MLP** μοντέλο αποδίδει εξαιρετικά στην ταξινόμηση της κυκλοφορίας ως **κανονική** ή **κακόβουλη**, επιτυγχάνοντας πολύ υψηλές τιμές σε όλες τις βασικές μετρικές και δείχνοντας ισχυρή ικανότητα γενίκευσης. Το **Traffic type** εκπαιδεύεται πολύ καλά στην ταξινόμηση των περισσότερων τύπων κίνησης, επιτυγχάνοντας υψηλά **F1-scores** για τις πλειοψηφικές κατηγορίες. Ωστόσο, παρουσιάζει κάποιες αδυναμίες στην αναγνώριση των κλάσεων με μικρότερο αριθμό δειγμάτων, για αυτό το ποσοστό ακρίβειας είναι μικρότερο.

Συνοπτικά Αποτελέσματα:

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	minibatch_kmeans_balanced_with_smote	Label	0.994753	0.993996	0.992617	0.994794	0.994031	0.992654
1	minibatch_kmeans_balanced_with_smote	Traffic Type	0.954483	0.946594	0.941131	0.956056	0.948577	0.943067

Label

>>> Training Report

	precision	recall	f1-score	support
Benign	1.00	0.96	0.98	781
Malicious	0.99	1.00	1.00	5750
accuracy			0.99	6531
macro avg	1.00	0.98	0.99	6531
weighted avg	0.99	0.99	0.99	6531

>>> Validation Report

	precision	recall	f1-score	support
Benign	0.99	0.96	0.97	260
Malicious	0.99	1.00	1.00	1918
accuracy			0.99	2178
macro avg	0.99	0.98	0.99	2178
weighted avg	0.99	0.99	0.99	2178

>>> Test Report

	precision	recall	f1-score	support
Benign	0.98	0.96	0.97	260
Malicious	0.99	1.00	1.00	1918
accuracy			0.99	2178
macro avg	0.99	0.98	0.98	2178
weighted avg	0.99	0.99	0.99	2178

Traffic Type

>>> Training Report

	precision	recall	f1-score	support
Audio	0.98	0.77	0.86	114
Background	0.00	0.00	0.00	19
Bruteforce	0.99	0.98	0.98	1431
DoS	0.98	0.97	0.97	1440
Information Gathering	0.95	0.99	0.97	1440
Mirai	0.93	0.96	0.95	1440
Text	0.92	0.86	0.89	125
Video	0.88	0.88	0.88	522
accuracy			0.96	6531
macro avg	0.83	0.80	0.81	6531
weighted avg	0.95	0.96	0.95	6531

>>> Validation Report

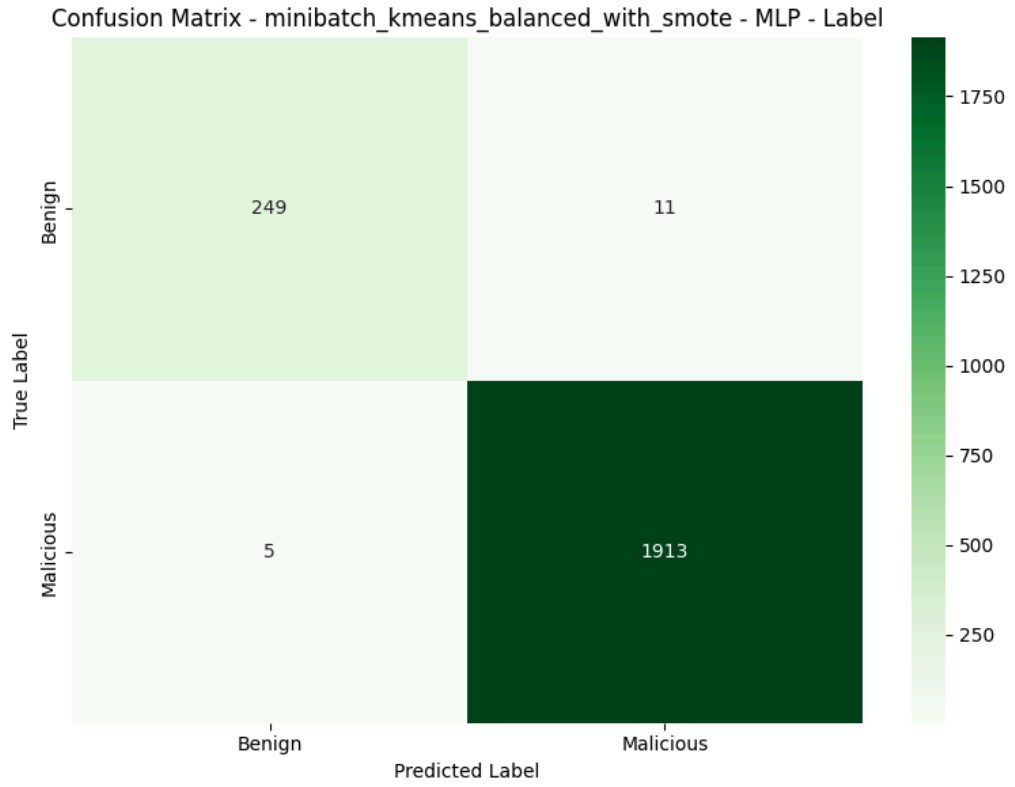
	precision	recall	f1-score	support
Audio	0.91	0.84	0.88	38
Background	0.00	0.00	0.00	6
Bruteforce	0.98	0.98	0.98	478
DoS	0.98	0.95	0.97	480
Information Gathering	0.96	0.99	0.97	480
Mirai	0.91	0.95	0.93	480
Text	0.89	0.60	0.71	42
Video	0.86	0.88	0.87	174
accuracy			0.95	2178
macro avg	0.81	0.77	0.79	2178
weighted avg	0.95	0.95	0.95	2178

>>> Test Report

	precision	recall	f1-score	support
Audio	0.90	0.74	0.81	38
Background	0.00	0.00	0.00	7
Bruteforce	0.98	0.97	0.98	477
DoS	0.97	0.95	0.96	480
Information Gathering	0.93	0.99	0.96	480
Mirai	0.92	0.93	0.93	480
Text	0.91	0.76	0.83	42
Video	0.88	0.88	0.88	174
accuracy			0.94	2178
macro avg	0.81	0.78	0.79	2178
weighted avg	0.94	0.94	0.94	2178

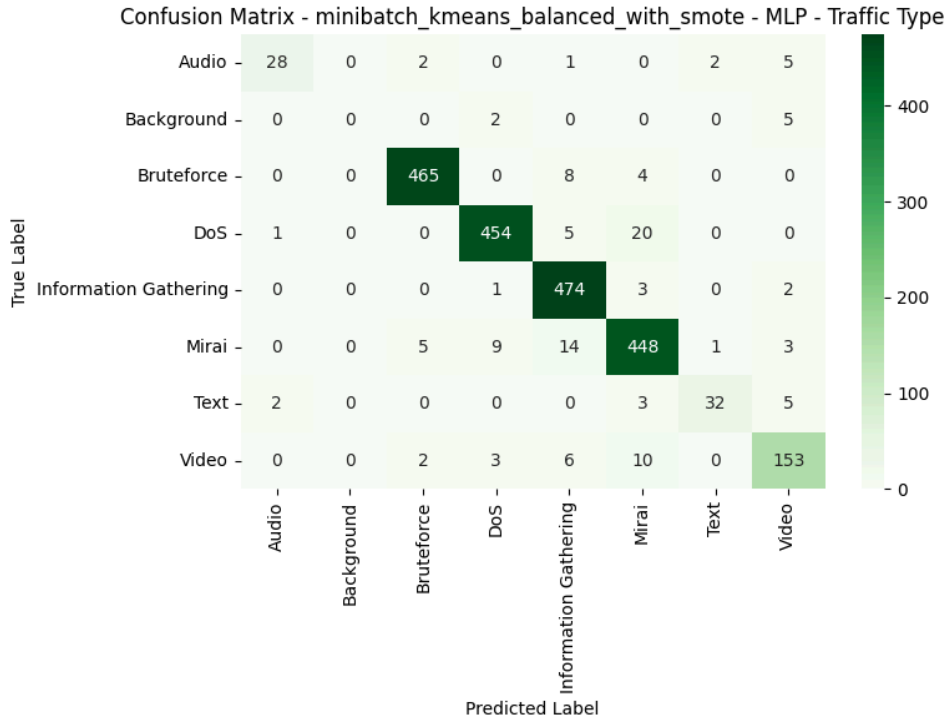
Πίνακας σύγχυσης του μοντέλου MLP

Label



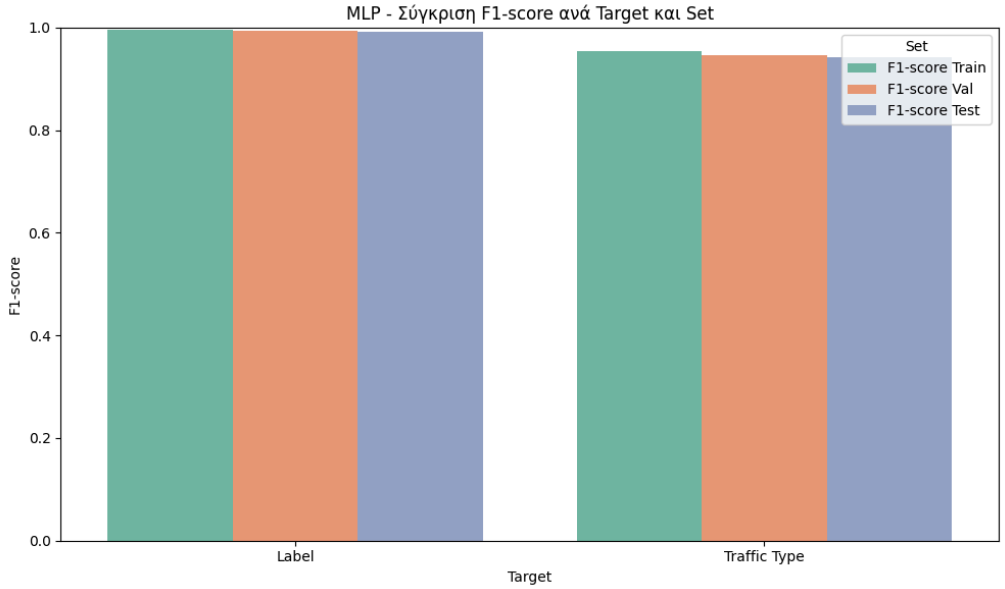
Αξιολογείται η απόδοση του **MLP** στην ταξινόμηση της κυκλοφορίας σε **κανονική** (Benign) και **κακόβουλη** (Malicious), αφού τα δεδομένα εξισορροπήθηκαν με **MiniBatch KMeans** και **SMOTE**. Το μοντέλο πέτυχε **πολύ υψηλή απόδοση**, με συνολικά 1913 **σωστά ταξινομημένες** κακόβουλες ροές και 249 σωστά ταξινομημένες κανονικές. Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα σφάλματα του μοντέλου: υπήρξαν 11 **False Positives**, δηλαδή 11 περιπτώσεις κανονικής κυκλοφορίας που ταξινομήθηκαν λανθασμένα ως κακόβουλες. Από την άλλη πλευρά, υπήρξαν μόνο 5 **False Negatives**, δηλαδή 5 κακόβουλες ροές που δεν ανιχνεύθηκαν. Επομένως, το μοντέλο παρουσιάζει πολύ καλή ισορροπία μεταξύ ακρίβειας και ευαισθησίας, κατάλληλο για συστήματα ανίχνευσης κακόβουλης δραστηριότητας σε δίκτυα.

Traffic Type



Οι διαγώνιες τιμές με πιο σκούρο πράσινο χρώμα αντιπροσωπεύουν τις **σωστές ταξινομήσεις** για κάθε τύπο κίνησης, ενώ οι εκτός διαγωνίου τιμές υποδεικνύουν εσφαλμένες ταξινομήσεις. Το μοντέλο παρουσιάζει ισχυρή απόδοση στην αναγνώριση κίνησης "**Bruteforce**", "**DoS**", "**Information Gathering**" και "**Mirai**", με μεγάλο αριθμό σωστών προβλέψεων για αυτές τις κατηγορίες. Ωστόσο, η κατηγορία "**Background**" εξαιτίας του πολύ μικρού αριθμού δειγμάτων της, αποτυγχάνει να ταξινομηθεί σωστά. Συγκεκριμένα, 2 δείγματα "**Background**" ταξινομήθηκαν λανθασμένα ως "**DoS**" και 5 δείγματα ταξινομήθηκαν λανθασμένα ως "**Mirai**". Επίσης, το μοντέλο παρουσιάζει μικρή σύγχυση μεταξύ ορισμένων κατηγοριών, όπως η εσφαλμένη ταξινόμηση 20 δειγμάτων "**DoS**" ως"**Mirai**" ή 14 δειγμάτων "**Mirai**" ως "**Information Gathering**". Επιπλέον, οι κατηγορίες "**Audio**" και "**Text**" έχουν λιγότερες σωστές ταξινομήσεις σε σύγκριση με άλλες, υποδεικνύοντας ότι το μοντέλο ενδέχεται να δυσκολεύεται με αυτούς τους συγκεκριμένους τύπους κίνησης.

Αναπαράσταση F1- scores

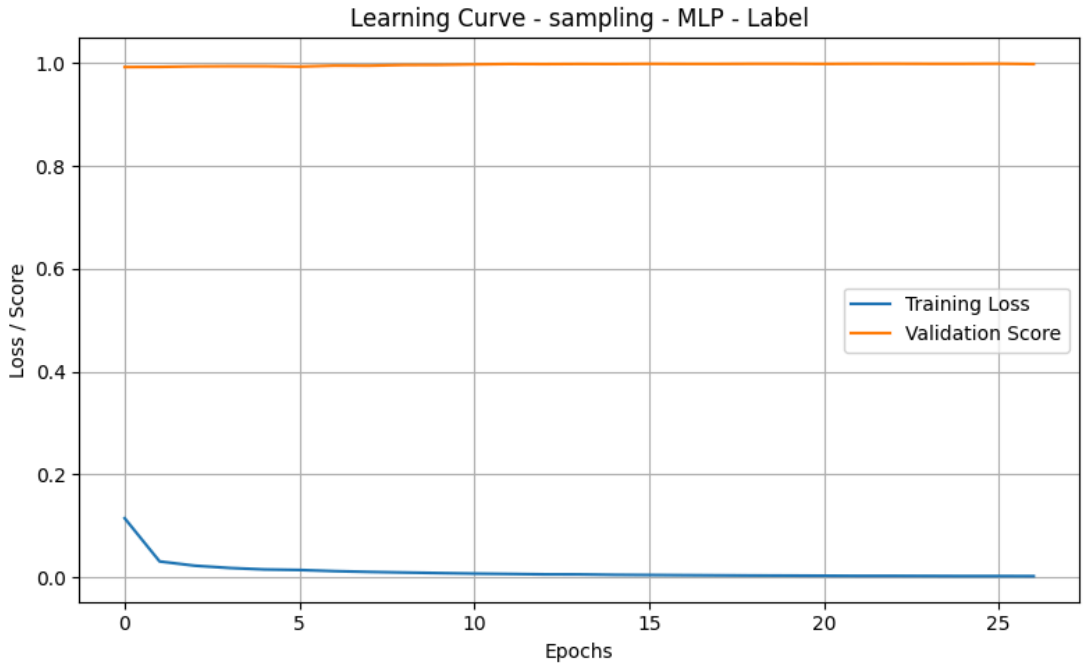


Το διάγραμμα συγκρίνει τις επιδόσεις του μοντέλου **MLP** με βάση το **F1-score** για δύο διαφορετικούς στόχους, "**Label**" και "**Traffic Type**", σε σύνολα **training**, **validation** και **testing**. Για την κατηγορία "**Label**", το MLP επιτυγχάνει σχεδόν τέλεια απόδοση (F1-score 1.0) σε όλα τα σύνολα, υποδεικνύοντας ότι το μοντέλο έχει μάθει να κατηγοριοποιεί άριστα τα δεδομένα για αυτόν τον στόχο και γενικεύει άψογα. Αντίθετα, για την κατηγορία "**Traffic Type**", ενώ η απόδοση παραμένει πολύ υψηλή (περίπου 0.96-0.97), είναι ελαφρώς χαμηλότερη από αυτή της "**Label**". Η μικρή διαφορά στα **F1-scores** μεταξύ των τριών συνόλων υποδηλώνει ότι το μοντέλο **MLP** γενικεύει εξαιρετικά καλά και δεν παρουσιάζει σημάδια **overfitting**.

➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε η Δειγματοληψία

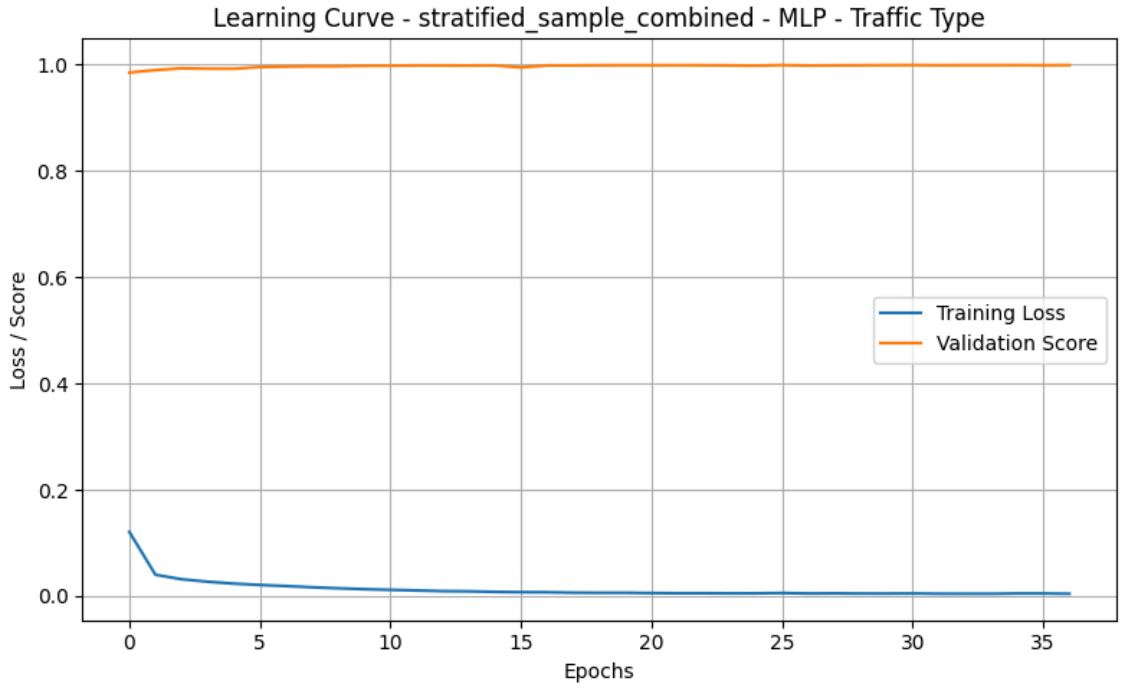
Καμπύλες μάθησης

Label



Τα αποτελέσματα έδειξαν **άριστη απόδοση**, με την ακρίβεια και το **F1-score** να αγγίζουν το 100% στο σύνολο επικύρωσης, ενώ η απώλεια στο σύνολο εκπαίδευσης μειώθηκε γρήγορα και σταθεροποιήθηκε σε πολύ χαμηλό επίπεδο. Η καμπύλη μάθησης επιβεβαιώνει τη σταθερή και αποτελεσματική σύγκλιση του μοντέλου, γεγονός που υποδηλώνει ότι οι κλάσεις είναι πιθανότατα καλά διαχωρίσιμες και το μοντέλο έχει προσαρμοστεί επαρκώς χωρίς σημάδια **overfitting**.

Traffic Type



Η εκπαίδευση του **MLP** μοντέλου για την πρόβλεψη της μεταβλητής **Traffic Type** ξεκίνησε με σχετικά υψηλή τιμή απώλειας στο σύνολο εκπαίδευσης, η οποία όμως μειώθηκε απότομα μέσα στα πρώτα λίγα epochs, υποδεικνύοντας **ταχεία προσαρμογή** του μοντέλου στα δεδομένα. Παράλληλα, η απόδοση στο σύνολο επικύρωσης ήταν ήδη πολύ υψηλή από την αρχή και σταθεροποιήθηκε γρήγορα κοντά στο 100%. Αυτή η συμπεριφορά δείχνει ότι το μοντέλο κατάφερε από νωρίς να μάθει αποτελεσματικά τα βασικά πρότυπα των δεδομένων, κάτι που πιθανόν οφείλεται στην καλή διαχωρισιμότητα των κατηγοριών του Traffic Type στα κύρια συστατικά του PCA.

Μετρικές Ταξινόμησης

Το **MLP** μοντέλο αποδίδει εξαιρετικά στην ταξινόμηση της κυκλοφορίας τόσο ως προς το "**Label**" όσο και προς το "**Traffic Type**", επιτυγχάνοντας πάρα πολύ υψηλές τιμές σε όλες τις βασικές μετρικές και δείχνοντας σχεδόν τέλεια ικανότητα γενίκευσης.

Συνοπτικά Αποτελέσματα:

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	SampledData	Label	0.999892	0.999849	0.999896	0.999893	0.999850	0.999896
1	SampledData	Traffic Type	0.998809	0.998553	0.998544	0.998809	0.998567	0.998556

Label

```
>>> Training Report
              precision    recall  f1-score   support

      Benign       1.00      0.98      0.99      2088
   Malicious       1.00      1.00      1.00    519327

    accuracy                           1.00    521415
   macro avg       1.00      0.99      0.99    521415
weighted avg       1.00      1.00      1.00    521415


>>> Validation Report
              precision    recall  f1-score   support

      Benign       1.00      0.97      0.98       696
   Malicious       1.00      1.00      1.00    173109

    accuracy                           1.00    173805
   macro avg       1.00      0.98      0.99    173805
weighted avg       1.00      1.00      1.00    173805


>>> Test Report
              precision    recall  f1-score   support

      Benign       1.00      0.98      0.99       696
   Malicious       1.00      1.00      1.00    173110

    accuracy                           1.00    173806
   macro avg       1.00      0.99      0.99    173806
weighted avg       1.00      1.00      1.00    173806
```

Traffic Type

```
>>> Training Report
                      precision    recall  f1-score   support

               Audio       0.95      0.80      0.87       522
          Background       0.73      1.00      0.84       522
          Bruteforce       0.99      0.98      0.99      2110
                 DoS       1.00      1.00      1.00    449455
Information Gathering       1.00      1.00      1.00     62302
               Mirai       0.98      0.97      0.97      5460
                Text       0.92      0.80      0.86       522
               Video       0.97      0.84      0.90       522

            accuracy                           1.00    521415
           macro avg       0.94      0.92      0.93    521415
        weighted avg       1.00      1.00      1.00    521415
```

```
>>> Validation Report
                      precision    recall  f1-score   support

               Audio       0.94      0.75      0.83       174
          Background       0.75      0.99      0.85       174
          Bruteforce       0.99      0.98      0.98       703
                 DoS       1.00      1.00      1.00    149819
Information Gathering       1.00      1.00      1.00     20767
               Mirai       0.98      0.96      0.97      1820
                Text       0.85      0.81      0.83       174
               Video       0.96      0.77      0.85       174

            accuracy                           1.00    173805
           macro avg       0.93      0.91      0.92    173805
        weighted avg       1.00      1.00      1.00    173805
```

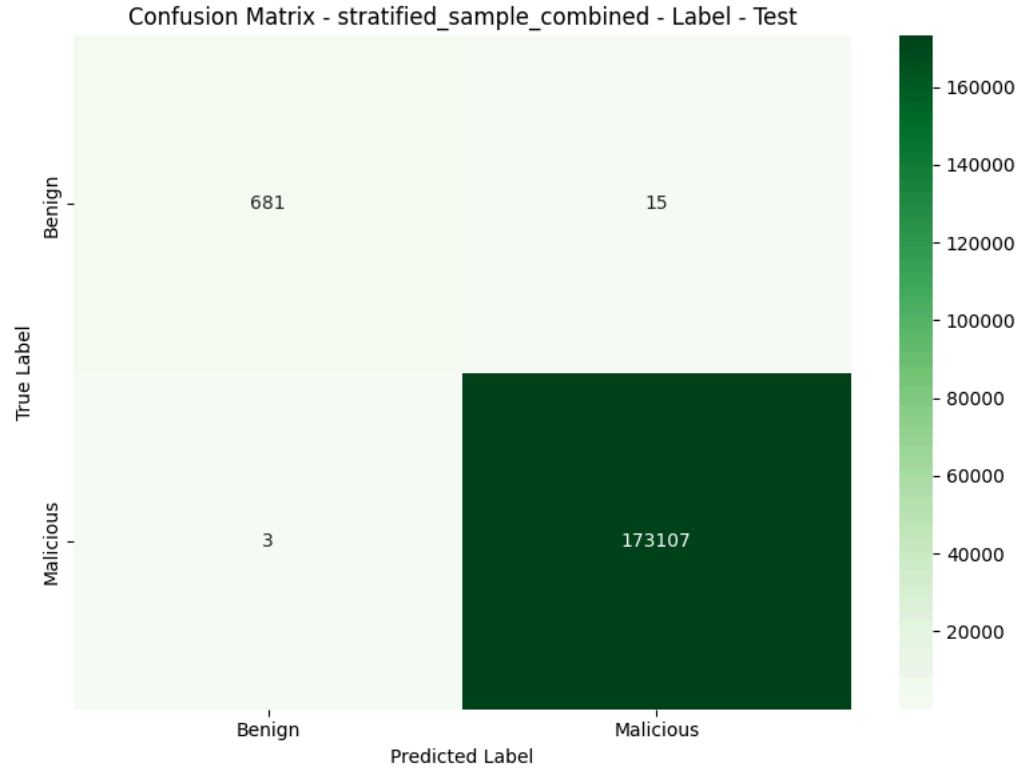
```
>>> Test Report
                      precision    recall  f1-score   support

               Audio       0.94      0.79      0.86       174
          Background       0.76      1.00      0.87       174
          Bruteforce       0.98      0.98      0.98       704
                 DoS       1.00      1.00      1.00    149819
Information Gathering       1.00      1.00      1.00     20767
               Mirai       0.98      0.96      0.97      1820
                Text       0.88      0.79      0.83       174
               Video       0.93      0.80      0.86       174

            accuracy                           1.00    173806
           macro avg       0.93      0.91      0.92    173806
        weighted avg       1.00      1.00      1.00    173806
```

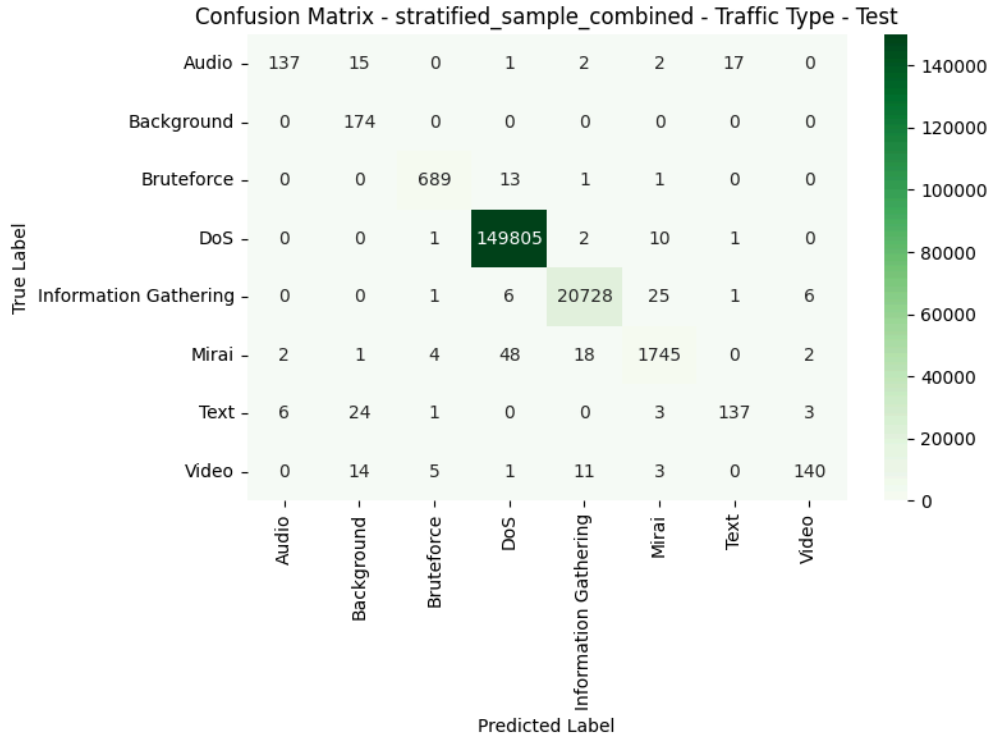
Πίνακας σύγχυσης του μοντέλου MLP

Label



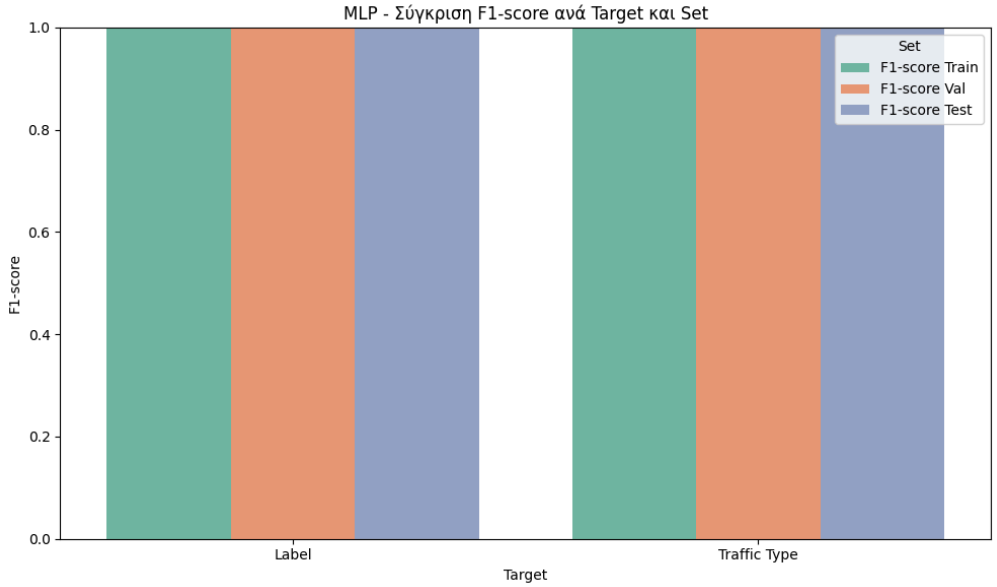
Το μοντέλο πέτυχε **πολύ υψηλή απόδοση**, με συνολικά 173107 **σωστά ταξινομημένες** κακόβουλες ροές και 681 σωστά ταξινομημένες κανονικές. Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα σφάλματα του μοντέλου: υπήρξαν 15 **False Positives**, δηλαδή 15 περιπτώσεις κανονικής κυκλοφορίας που ταξινομήθηκαν λανθασμένα ως κακόβουλες. Από την άλλη πλευρά, υπήρξαν μόνο 3 **False Negatives**, δηλαδή 3 κακόβουλες ροές που δεν ανιχνεύθηκαν. Επομένως, το μοντέλο παρουσιάζει πολύ καλή ισορροπία μεταξύ ακρίβειας και ευαισθησίας, κατάλληλο για συστήματα ανίχνευσης κακόβουλης δραστηριότητας σε δίκτυα.

Traffic Type



Οι διαγώνιες τιμές με πιο σκούρο πράσινο χρώμα αντιπροσωπεύουν τις **σωστές ταξινομήσεις** για κάθε τύπο κίνησης, ενώ οι εκτός διαγωνίου τιμές υποδεικνύουν εσφαλμένες ταξινομήσεις. Το μοντέλο παρουσιάζει ισχυρή απόδοση στην αναγνώριση κίνησης "**DoS**" και "**Information Gathering**" με μεγάλο αριθμό σωστών προβλέψεων για αυτές τις κατηγορίες. Εδώ, η κατηγορία "**Background**" επιτυγχάνει να ταξινομηθεί σωστά. Επίσης, το μοντέλο παρουσιάζει μικρή σύγχυση μεταξύ ορισμένων κατηγοριών - όμως σε μικρό αριθμό δειγμάτων, όπως η εσφαλμένη ταξινόμηση 48 δειγμάτων "**Mirai**" ως "**DoS**". Επιπλέον, κατηγορίες όπως "**Audio**" και "**Video**" έχουν λιγότερες σε αριθμό σωστές ταξινομήσεις σε σύγκριση με άλλες, υποδεικνύοντας ότι το μοντέλο ενδέχεται να δυσκολεύεται με αυτούς τους συγκεκριμένους τύπους κίνησης.

Αναπαράσταση F1- scores

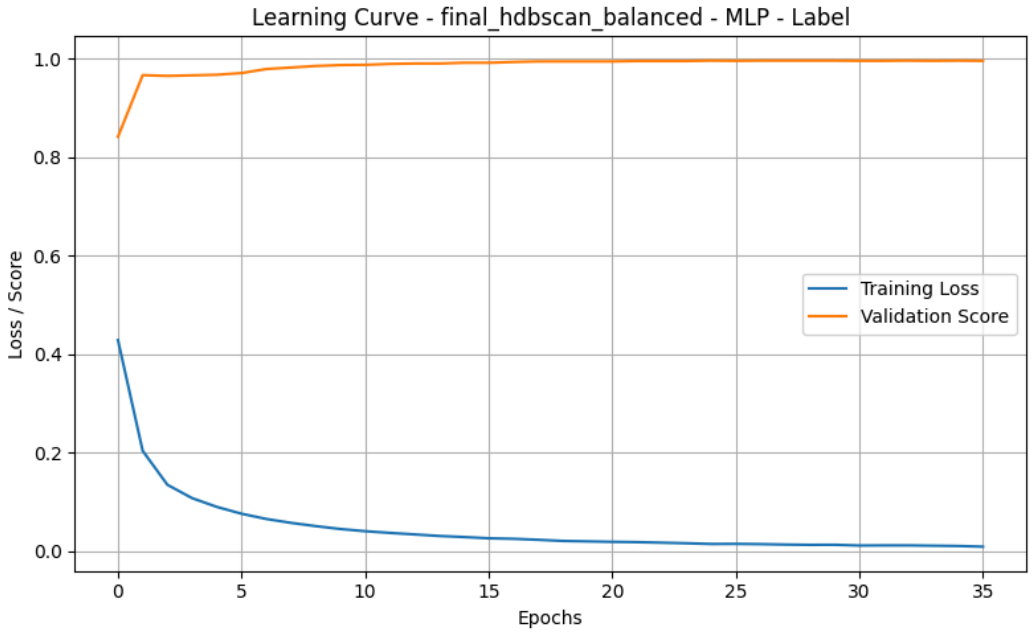


Το διάγραμμα συγκρίνει τις επιδόσεις του μοντέλου **MLP** με βάση το **F1-score** για δύο διαφορετικούς στόχους, "**Label**" και "**Traffic Type**", σε σύνολα **training**, **validation** και **testing**. Τόσο για την κατηγορία "**Label**" όσο και την "**Traffic Type**", το **MLP** επιτυγχάνει σχεδόν τέλεια απόδοση (0.99 και πάνω) σε όλα τα σύνολα, υποδεικνύοντας ότι το μοντέλο έχει μάθει να κατηγοριοποιεί άριστα τα δεδομένα για αυτόν τον στόχο και γενικεύει άψογα.

➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε ο HDBSCAN

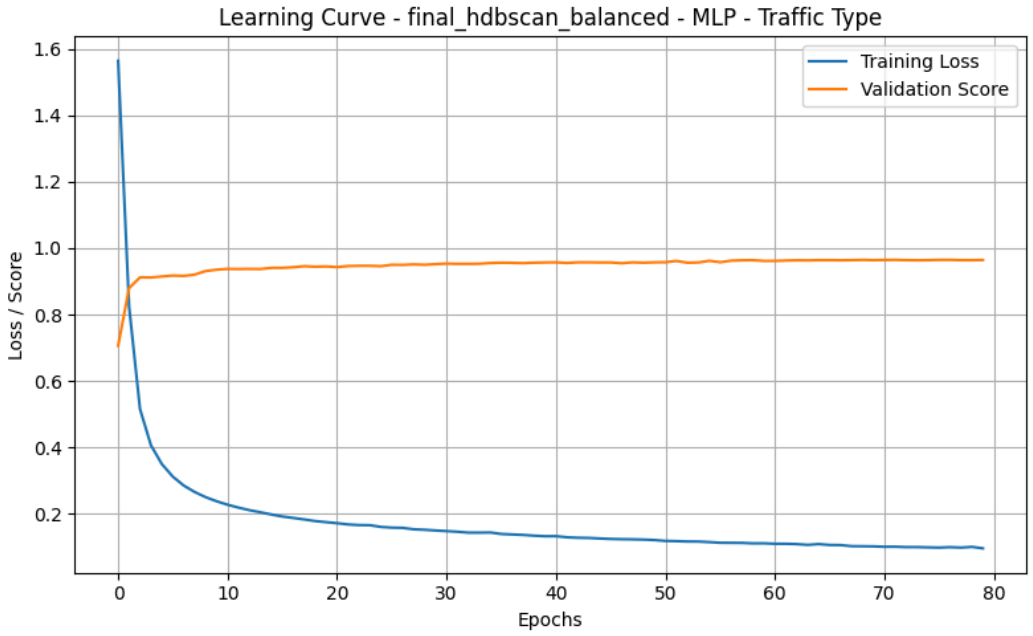
Καμπύλες μάθησης

Label



Το διάγραμμα απόδοσης, που αναπαριστά την καμπύλη μάθησης, δείχνει ότι η καμπύλη του **Training Loss** μειώνεται σταθερά με την αύξηση των εποχών, υποδεικνύοντας ότι το μοντέλο μαθαίνει από τα δεδομένα εκπαίδευσης. Παράλληλα, το **Validation Score** αυξάνεται γρήγορα στις πρώτες εποχές και στη συνέχεια σταθεροποιείται σε υψηλά επίπεδα (κοντά στο 1.0), δείχνοντας ότι το μοντέλο γενικεύει καλά σε ξένα δεδομένα και δεν παρατηρείται **overfitting**. Αυτή η συμπεριφορά υποδηλώνει μια αποτελεσματική εκπαίδευση του μοντέλου **MLP**.

Traffic Type



Η καμπύλη εκμάθησης για το μοντέλο **MLP** με στόχο πρόβλεψης το **Traffic Type** δείχνει ότι το μοντέλο εκπαιδεύεται αποτελεσματικά αλλά η στασιμότητα του **validation score** σε συνδυασμό με την συνεχώς μειούμενη **training loss** δείχνει ότι έχει φτάσει το μέγιστο δυναμικό του, και περαιτέρω εκπαίδευση δεν προσφέρει ουσιαστική βελτίωση . Συγκεκριμένα, το **training loss** μειώνεται σταθερά και φτάνει σε πολύ χαμηλά επίπεδα, γεγονός που υποδηλώνει ότι το μοντέλο μαθαίνει καλά τα δεδομένα εκπαίδευσης. Ωστόσο, η καμπύλη του **validation score** σταθεροποιείται σχετικά νωρίς γύρω στο **0.96** χωρίς σημαντική βελτίωση καθώς αυξάνονται τα epochs. Αυτή η απόκλιση μεταξύ εκπαίδευσης και επικύρωσης δείχνει ότι το μοντέλο έχει μάθει πολύ καλά τα δεδομένα εκπαίδευσης, αλλά δεν γενικεύει εξίσου καλά στα δεδομένα **validation**.

Μετρικές Ταξινόμησης

Το **MLP** μοντέλο αποδίδει εξαιρετικά στην ταξινόμηση της κυκλοφορίας ως προς το “**Label**” και ικανοποιητικά ως προς το “**Traffic Type**”, επιτυγχάνοντας γενικά υψηλές τιμές σε όλες τις βασικές μετρικές και δείχνοντας καλή γενίκευση.

Συνοπτικά Αποτελέσματα:

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	final_hdbscan_balanced	Label	0.996206	0.993971	0.994194	0.996206	0.993961	0.994193
1	final_hdbscan_balanced	Traffic Type	0.966196	0.959967	0.963168	0.968484	0.963066	0.965854

Label

```
>>> Training Report
              precision    recall  f1-score   support

      Benign       0.99      0.99      0.99      2622
   Malicious       1.00      1.00      1.00     10292

    accuracy                           1.00     12914
   macro avg       0.99      0.99      0.99     12914
weighted avg       1.00      1.00      1.00     12914


>>> Validation Report
              precision    recall  f1-score   support

      Benign       0.98      0.99      0.99       874
   Malicious       1.00      1.00      1.00      3431

    accuracy                           0.99      4305
   macro avg       0.99      0.99      0.99      4305
weighted avg       0.99      0.99      0.99      4305


>>> Test Report
              precision    recall  f1-score   support

      Benign       0.99      0.99      0.99       874
   Malicious       1.00      1.00      1.00      3431

    accuracy                           0.99      4305
   macro avg       0.99      0.99      0.99      4305
weighted avg       0.99      0.99      0.99      4305
```

Traffic Type

```
>>> Training Report
                      precision    recall  f1-score   support

               Audio       0.95      0.83      0.89       300
          Background       0.66      0.39      0.49       300
          Bruteforce       0.98      0.80      0.88        54
                 DoS       1.00      0.99      1.00      7370
Information Gathering       0.99      0.99      0.99      2718
               Mirai       0.90      0.92      0.91       150
                Text       0.92      0.79      0.85       300
               Video       0.88      0.98      0.93      1722

            accuracy                           0.97     12914
           macro avg       0.91      0.84      0.87     12914
        weighted avg       0.97      0.97      0.97     12914
```

```
>>> Validation Report
                      precision    recall  f1-score   support

               Audio       0.90      0.78      0.83       100
          Background       0.65      0.35      0.45       100
          Bruteforce       1.00      0.56      0.71        18
                 DoS       0.99      1.00      0.99      2457
Information Gathering       0.98      0.99      0.99       906
               Mirai       0.86      0.84      0.85        50
                Text       0.87      0.76      0.81       100
               Video       0.87      0.97      0.92       574

            accuracy                           0.96      4305
           macro avg       0.89      0.78      0.82      4305
        weighted avg       0.96      0.96      0.96      4305
```

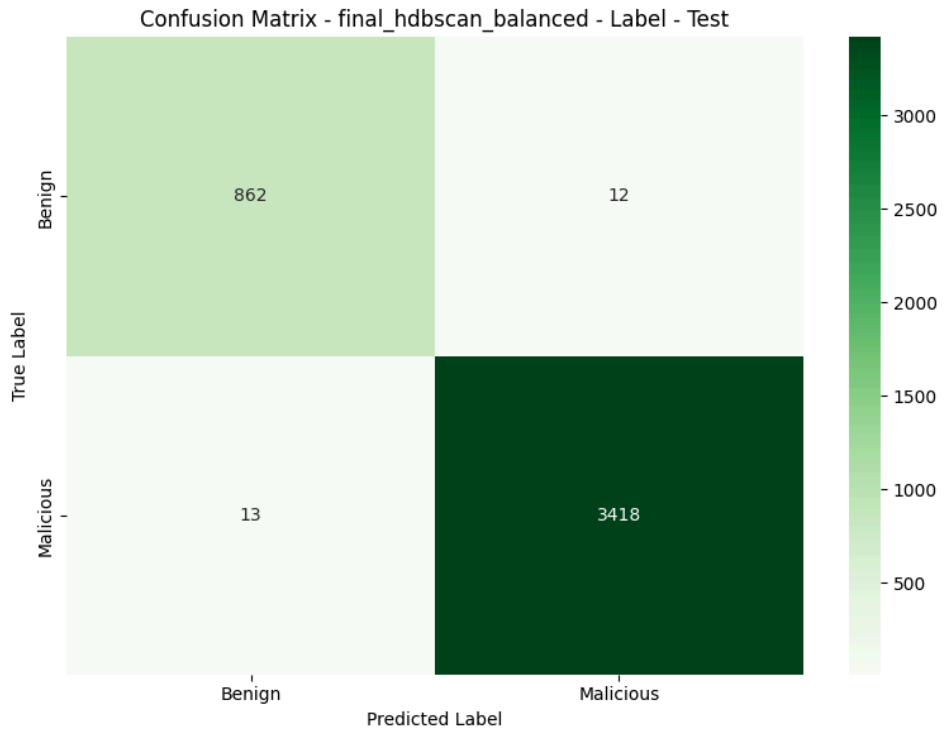
```
>>> Test Report
                      precision    recall  f1-score   support

               Audio       0.90      0.81      0.85       100
          Background       0.64      0.36      0.46       100
          Bruteforce       0.86      0.67      0.75        18
                 DoS       0.99      0.99      0.99      2457
Information Gathering       0.99      1.00      0.99       906
               Mirai       0.92      0.94      0.93        50
                Text       0.95      0.79      0.86       100
               Video       0.87      0.97      0.92       574

            accuracy                           0.97      4305
           macro avg       0.89      0.82      0.85      4305
        weighted avg       0.96      0.97      0.96      4305
```

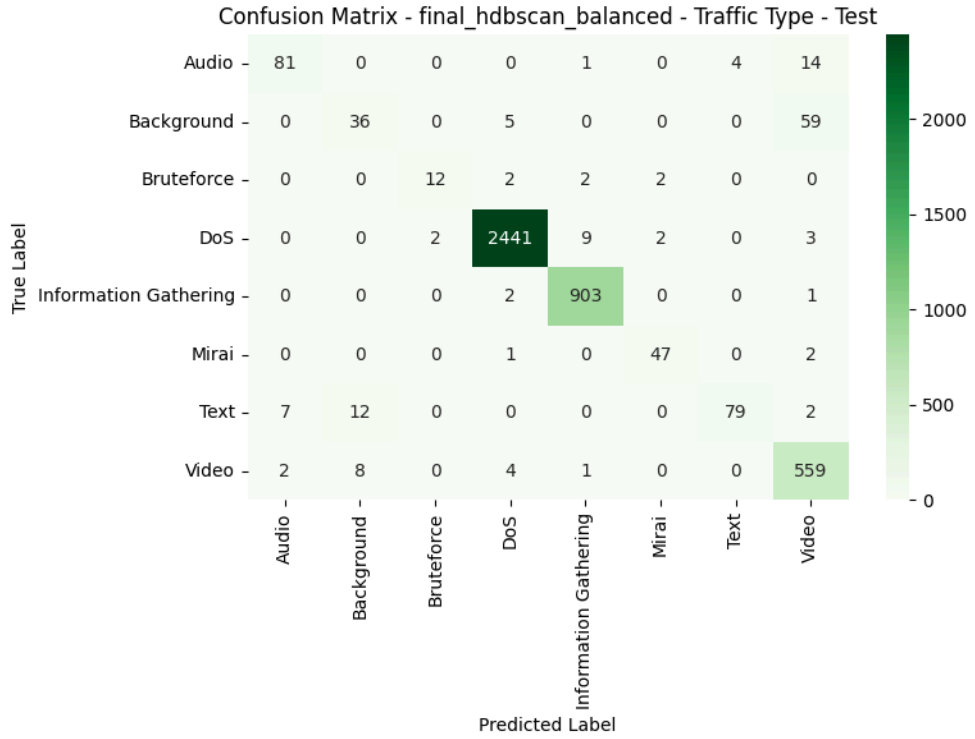
Πίνακας σύγχυσης του μοντέλου MLP

Label



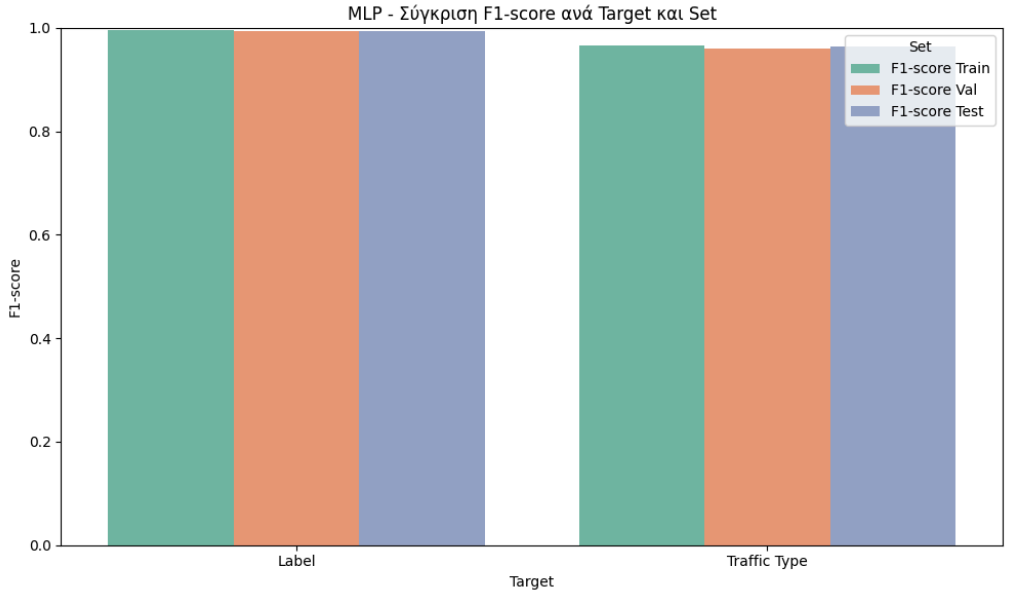
Το παρόν Confusion Matrix αναδεικνύει την ισχυρή απόδοση του μοντέλου στην ταξινόμηση δεδομένων του συνόλου δοκιμής στην κατηγορία "**Label**", διαχωρίζοντας αποτελεσματικά μεταξύ "**Benign**" και "**Malicious**" περιπτώσεων. Ειδικότερα, το μοντέλο επέδειξε υψηλή ακρίβεια στην αναγνώριση 3418 "Malicious" περιστατικών (**True Positive)** και 862 "Benign" περιστατικών (**True Negatives)**. Παράλληλα, ο αριθμός των λανθασμένων ταξινομήσεων είναι εξαιρετικά χαμηλός, με μόλις 12 "**Benign**" περιπτώσεις να χαρακτηρίζονται εσφαλμένα ως "**Malicious**" (**False Positives**) και 13 "**Malicious**" περιπτώσεις να εκλαμβάνονται λανθασμένα ως "**Benign**" (**False Negatives**)

Traffic Type



Το μοντέλο παρουσιάζει ισχυρή απόδοση στις κατηγορίες **DoS**, **Information Gathering** και **Video**, με 2441, 903 και 559 σωστά ταξινομημένα δείγματα αντίστοιχα.Ωστόσο, παρατηρούνται μικρές συγχύσεις μεταξύ άλλων κατηγοριών, τύπους όπως το "**Background**" όπου 59 δείγματα ως “**Video**”. Η κατηγορία **Background**, λόγω του πολύ μικρού αριθμού δειγμάτων της, παρουσιάζει αδυναμία ταξινόμησης, με τα δείγματα της να αποδίδονται σε λάθος κατηγορίες. Παρόμοια, οι κατηγορίες **Audio** και **Text** έχουν λιγότερες σωστές προβλέψεις, υποδεικνύοντας ότι το μοντέλο δυσκολεύεται να τις διακρίνει. Παρόλα αυτά, η συνολική εικόνα είναι θετική, υποδηλώνοντας ότι το μοντέλο είναι αποτελεσματικό στην αναγνώριση των κυρίαρχων χαρακτηριστικών κάθε τύπου κίνησης, με περιθώριο για περαιτέρω βελτιστοποίηση.

Αναπαράσταση F1- scores



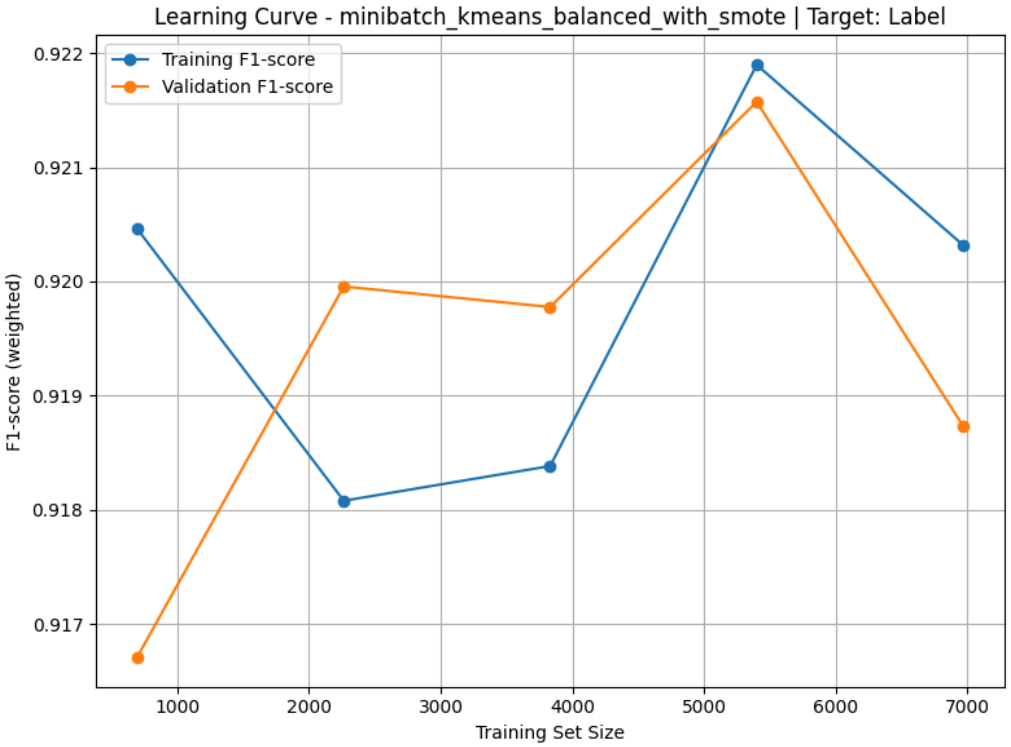
Το barplot συγκρίνει το **F1-score** του μοντέλου **MLP** για τους στόχους "**Label**" και "**Traffic Type**" σε σύνολα εκπαίδευσης, επικύρωσης και δοκιμής. Για τον στόχο "**Label**", το **F1-score** είναι σχεδόν τέλειο (1.0) και σταθερό σε όλα τα σύνολα, υποδεικνύοντας άριστη γενίκευση. Για τον στόχο "**Traffic Type**", το **F1-score** παραμένει εξαιρετικά υψηλό (κοντά στο 0.98-0.99) και συνεπές σε όλα τα σύνολα. Συνολικά, το διάγραμμα καταδεικνύει την ισχυρή και αξιόπιστη απόδοση του μοντέλου **MLP** και στους δύο στόχους ταξινόμησης, χωρίς ενδείξεις υπερ-προσαρμογής ή υπο-προσαρμογής.

Εκπαίδευση κατηγοριοποιητή βασισμένο σε SVM

➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε ο MiniBatchKmeans

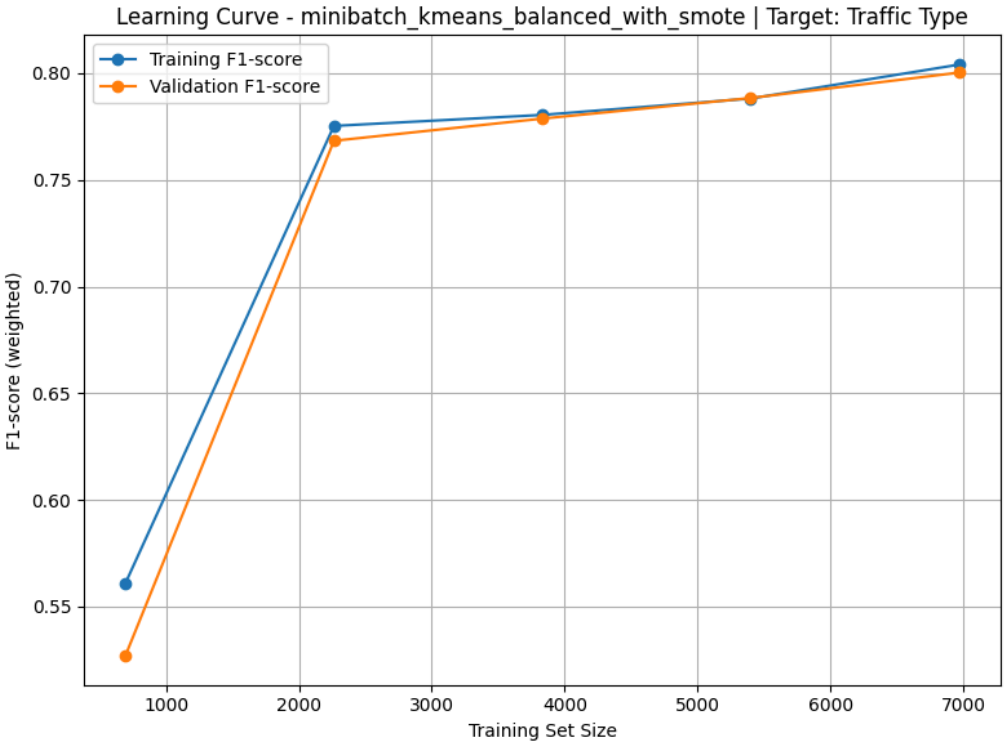
Καμπύλες μάθησης

Label



Η καμπύλη μάθησης παρουσιάζει διακυμάνσεις, αλλά τα αναλυτικά αποτελέσματα του πίνακα επιβεβαιώνουν ότι το μοντέλο γενικεύει καλά, αφού το **F1-score** στο **validation** (0.931214) είναι υψηλότερο από αυτό του **training** (0.918584), και το **F1-score** στο **test set** (0.922556) παραμένει σε υψηλά επίπεδα, κοντά στο training. Η βελτίωση στην απόδοση της μικρότερης κλάσης **Benign** στο **validation set**, σε σύγκριση με το **training**, είναι ένα θετικό σημάδι ότι το μοντέλο μαθαίνει να αναγνωρίζει και αυτή την κλάση αποτελεσματικά, ακόμα κι αν η συνολική βαρύτητα των αποτελεσμάτων επηρεάζεται από την κυρίαρχη κλάση **Malicious**.

Traffic Type



Παρότι το μοντέλο εμφάνισε σταθερή και συγκλίνουσα συμπεριφορά στις καμπύλες μάθησης, με **F1-score** και για το **training** και για το **validation** κοντά στο **0.80**, η τελική ακρίβεια στο πρόβλημα ταξινόμησης του **Traffic Type** παρέμεινε σχετικά χαμηλότερη σε σύγκριση με το αντίστοιχο πρόβλημα ταξινόμησης **Label**. Αυτό οφείλεται κυρίως στην πολυπλοκότητα της ταξινόμησης σε 8 διαφορετικές κατηγορίες, καθώς και στη σύγχυση μεταξύ παρόμοιων τύπων κίνησης (όπως Video και Mirai). Παρ' όλα αυτά, το μοντέλο δεν παρουσίασε ενδείξεις overfitting, γεγονός που φανερώνει καλή ικανότητα γενίκευσης.

Μετρικές Ταξινόμησης

Το **SVM** μοντέλο απέδωσε ιδιαίτερα καλά στην ταξινόμηση μεταξύ κανονικής και κακόβουλης κυκλοφορίας (**Label**), επιτυγχάνοντας υψηλά ποσοστά ακρίβειας και **F1-score** σε όλα τα υποσύνολα δεδομένων (Train, Validation, Test). Από την άλλη πλευρά, στην πιο απαιτητική ταξινόμηση κατά τύπο κυκλοφορίας (**Traffic Type**), το **SVM** εμφανίζει ικανοποιητική αλλά χαμηλότερη απόδοση, με **F1-score** Test περίπου στο **0.794**. Αυτό οφείλεται κυρίως στη δυσκολία του μοντέλου να χειριστεί ορθά τις λιγότερο συχνές κατηγορίες, κάτι που αντικατοπτρίζεται και στη μικρή πτώση των επιδόσεων σε σχέση με τις πλειοψηφικές κλάσεις.

Συνοπτικά Αποτελέσματα:

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	minibatch_kmeans_balanced_with_smote	Label	0.918584	0.931214	0.922556	0.917164	0.930211	0.921488
1	minibatch_kmeans_balanced_with_smote	Traffic Type	0.802528	0.800103	0.794171	0.802174	0.798898	0.795684

Label

```
>>> Training Report
              precision    recall  f1-score   support

      Benign       0.64      0.70      0.67       781
   Malicious       0.96      0.95      0.95      5750

    accuracy                           0.92      6531
   macro avg       0.80      0.82      0.81      6531
weighted avg       0.92      0.92      0.92      6531

>>> Validation Report
              precision    recall  f1-score   support

      Benign       0.69      0.74      0.72       260
   Malicious       0.96      0.96      0.96      1918

    accuracy                           0.93      2178
   macro avg       0.83      0.85      0.84      2178
weighted avg       0.93      0.93      0.93      2178

>>> Test Report
              precision    recall  f1-score   support

      Benign       0.66      0.70      0.68       260
   Malicious       0.96      0.95      0.96      1918

    accuracy                           0.92      2178
   macro avg       0.81      0.83      0.82      2178
weighted avg       0.92      0.92      0.92      2178
```

Traffic Type

>>> Training Report

	precision	recall	f1-score	support
Audio	0.74	0.77	0.76	114
Background	0.13	0.95	0.22	19
Bruteforce	0.96	0.86	0.91	1431
DoS	0.85	0.87	0.86	1440
Information Gathering	0.77	0.95	0.85	1440
Mirai	0.72	0.66	0.69	1440
Text	0.85	0.78	0.81	125
Video	0.74	0.45	0.56	522
accuracy			0.80	6531
macro avg	0.72	0.79	0.71	6531
weighted avg	0.82	0.80	0.80	6531

>>> Validation Report

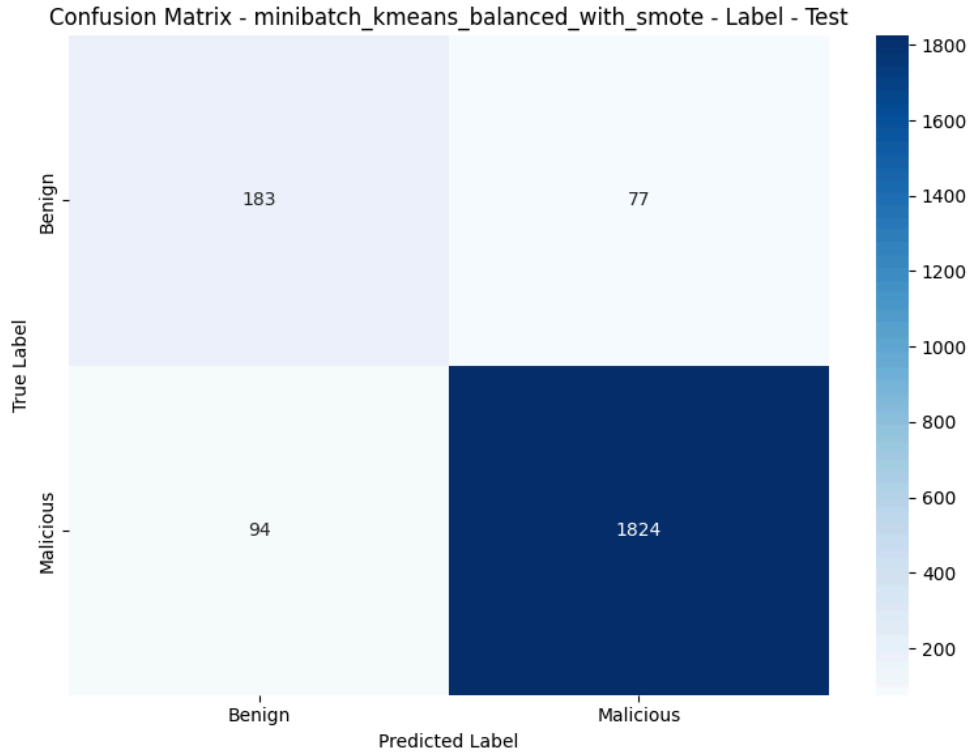
	precision	recall	f1-score	support
Audio	0.82	0.82	0.82	38
Background	0.10	1.00	0.18	6
Bruteforce	0.96	0.88	0.92	478
DoS	0.86	0.87	0.87	480
Information Gathering	0.78	0.95	0.86	480
Mirai	0.72	0.64	0.68	480
Text	0.77	0.64	0.70	42
Video	0.69	0.41	0.52	174
accuracy			0.80	2178
macro avg	0.71	0.78	0.69	2178
weighted avg	0.81	0.80	0.80	2178

>>> Test Report

	precision	recall	f1-score	support
Audio	0.69	0.71	0.70	38
Background	0.13	0.71	0.22	7
Bruteforce	0.96	0.87	0.91	477
DoS	0.85	0.85	0.85	480
Information Gathering	0.77	0.97	0.85	480
Mirai	0.69	0.63	0.66	480
Text	0.89	0.79	0.84	42
Video	0.72	0.44	0.54	174
accuracy			0.80	2178
macro avg	0.71	0.75	0.70	2178
weighted avg	0.81	0.80	0.79	2178

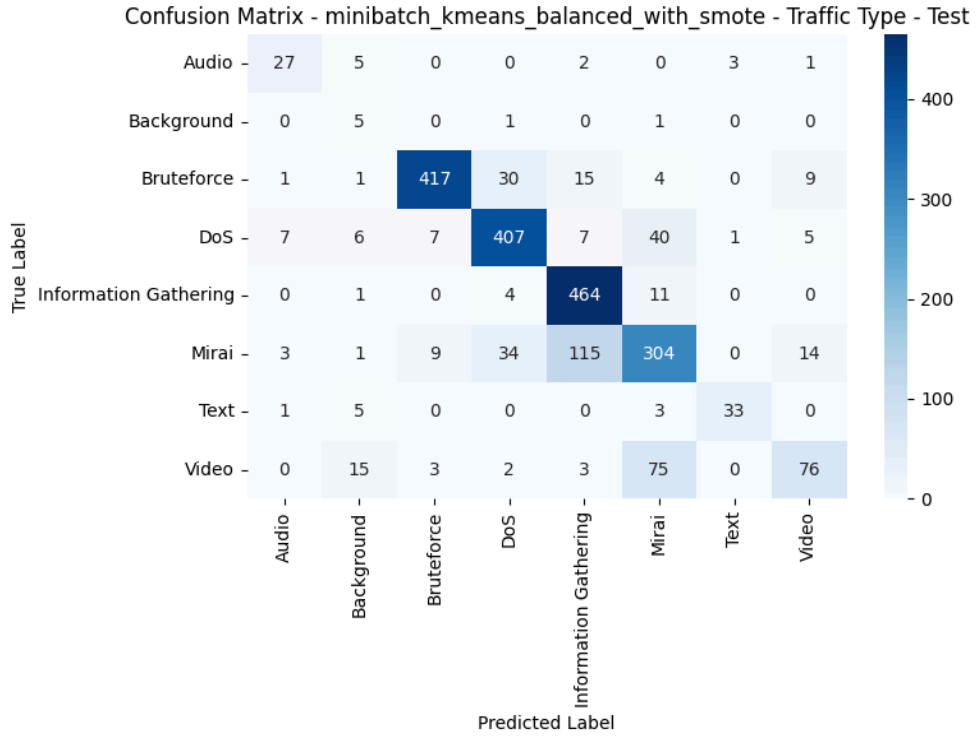
Πίνακας σύγχυσης του μοντέλου MLP

Label



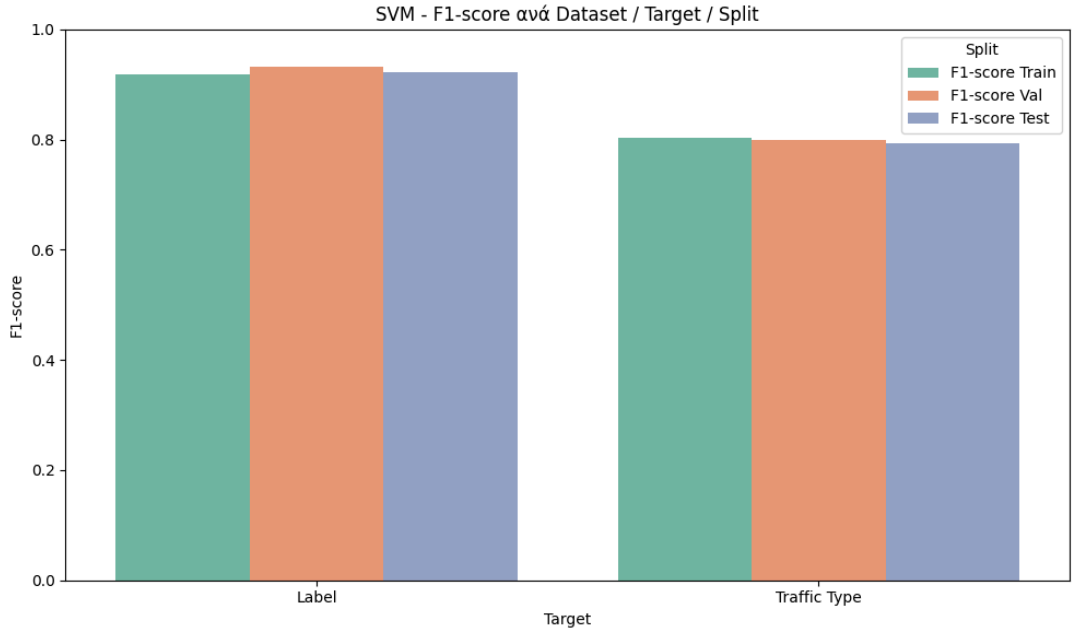
Αξιολογείται η απόδοση του **SVM** στην ταξινόμηση της κυκλοφορίας σε **κανονική** (Benign) και **κακόβουλη** (Malicious), αφού τα δεδομένα εξισορροπήθηκαν με **MiniBatch KMeans** και **SMOTE**. Το μοντέλο πέτυχε πολύ υψηλή απόδοση, με **1824 σωστά ταξινομημένες** κακόβουλες ροές και **183** σωστά ταξινομημένες κανονικές. Τα σφάλματα εντοπίζονται σε **77** περιπτώσεις **False Positives** δηλαδή κανονικές ροές που αναγνωρίστηκαν λανθασμένα ως κακόβουλες και 94 **False Negatives** δηλαδή κακόβουλες ροές που δεν ανιχνεύθηκαν. Παρά τα σφάλματα, η συνολική επίδοση παραμένει εξαιρετική, με καλή ισορροπία μεταξύ ακρίβειας και ευαισθησίας, καθιστώντας το μοντέλο κατάλληλο για χρήση σε συστήματα ανίχνευσης.

Traffic Type



Οι διαγώνιες τιμές με πιο σκούρο χρώμα αντιπροσωπεύουν τις σωστές ταξινομήσεις για κάθε τύπο κίνησης, ενώ οι εκτός διαγωνίου τιμές υποδεικνύουν τις εσφαλμένες. Το μοντέλο παρουσιάζει ισχυρή απόδοση στις κατηγορίες **Bruteforce**, **DoS**, **Information Gathering** και **Mirai**, με 417, 407, 464 και 304 σωστά ταξινομημένα δείγματα αντίστοιχα. Ωστόσο, παρατηρούνται σημαντικά σφάλματα στην κατηγορία **Video**, όπου το μοντέλο μπερδεύει 75 δείγματα ως "**Mirai**", και στην κατηγορία **Mirai**, με 115 δείγματα να ταξινομούνται λανθασμένα ως "**Information Gathering** ". Η κατηγορία **Background**, λόγω του πολύ μικρού αριθμού δειγμάτων της, παρουσιάζει αδυναμία ταξινόμησης, με τα δείγματα της να αποδίδονται σε λάθος κατηγορίες. Παρόμοια, οι κατηγορίες **Audio** και **Text** έχουν λιγότερες σωστές προβλέψεις, υποδεικνύοντας ότι το μοντέλο δυσκολεύεται να τις διακρίνει. Γενικά, το μοντέλο εμφανίζει **καλή συνολική επίδοση**.

Αναπαράσταση F1- scores

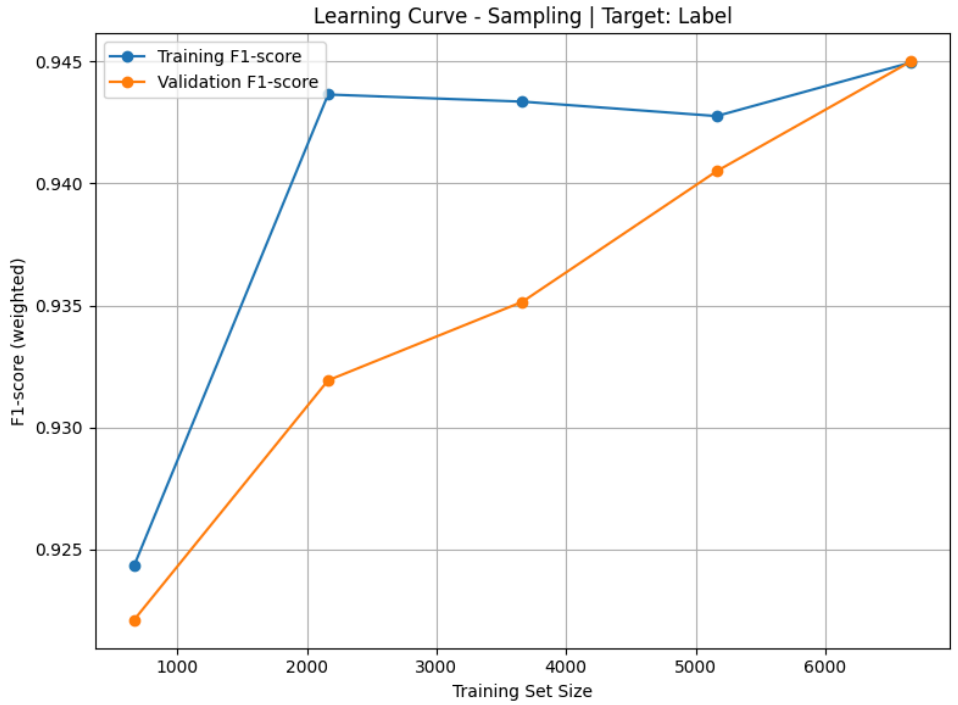


Το διάγραμμα απεικονίζει τις επιδόσεις του μοντέλου **SVM** σε σχέση με το **F1-score** για τις κλάσεις "**Label**" και "**Traffic Type**", διαχωρισμένες σε σύνολα **εκπαίδευσης** (Train), **επικύρωσης** (Val) και **ελέγχου** (Test). Για την κατηγορία "**Label**", παρατηρείται εξαιρετικά υψηλό **F1-score**, το οποίο κυμαίνεται περίπου στο **0.92-0.93** σε όλα τα σύνολα, με το **Validation set** να εμφανίζει ελαφρώς την καλύτερη επίδοση, υποδεικνύοντας ισχυρή γενίκευση του μοντέλου. Αντίθετα, για την κατηγορία "**Traffic Type**", τα **F1-scores** είναι χαμηλότερα, κινούμενα περίπου στο **0.80** για όλα τα σύνολα, αν και παραμένουν σε ικανοποιητικά επίπεδα. Η σχετική ομοιομορφία των **F1-scores** μεταξύ των τριών συνόλων και για τις δύο κατηγορίες υποδηλώνει ότι το μοντέλο δεν κάνει **overfitting** και γενικεύει καλά στα καινούρια δεδομένα, παρά τη διαφορά στην απόλυτη τιμή του F1-score μεταξύ των δύο στόχων.

➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε η Δειγματοληψία

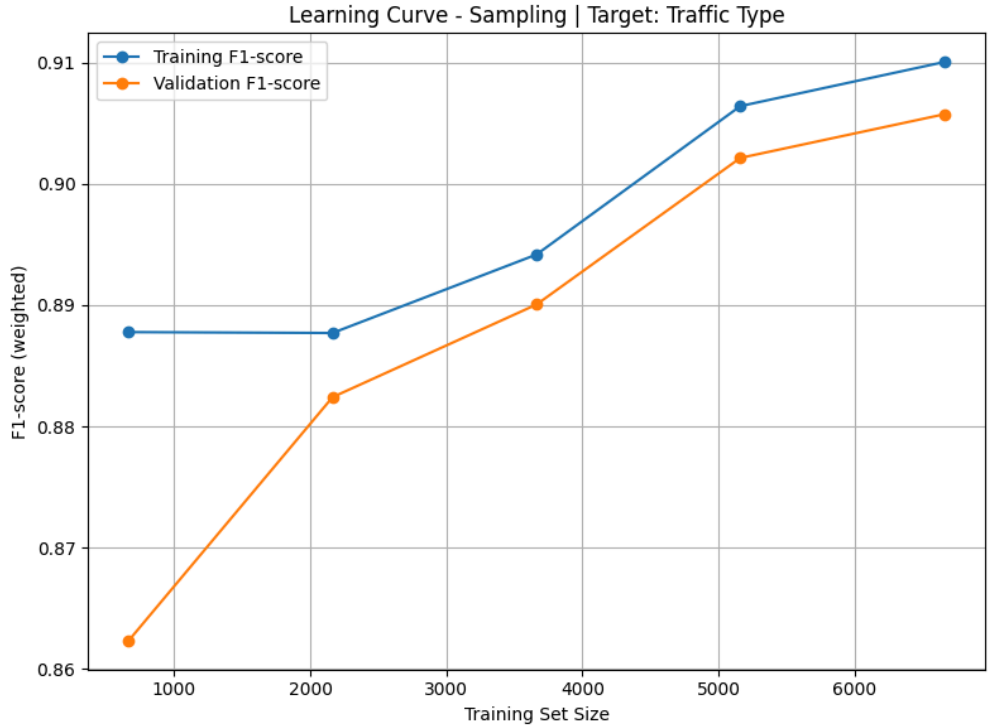
Καμπύλες μάθησης

Label



Η καμπύλη **εκπαίδευσης** ξεκινά από υψηλό **F1-score** (περίπου 0.925) και παραμένει σε υψηλά επίπεδα, με διακυμάνσεις, φτάνοντας σχεδόν το 0.945. Αντίστοιχα, η καμπύλη **επικύρωσης** ξεκινά χαμηλότερα (περίπου 0.91), αλλά παρουσιάζει σταθερή ανοδική πορεία, συγκλίνοντας προς την καμπύλη εκπαίδευσης και φτάνοντας περίπου το 0.945 στο μεγαλύτερο μέγεθος συνόλου εκπαίδευσης. Η σύγκλιση των δύο καμπυλών σε υψηλές τιμές υποδηλώνει ότι το μοντέλο γενικεύει καλά και επωφελείται από περισσότερα δεδομένα εκπαίδευσης, χωρίς εμφανή σημάδια υπερ-προσαρμογής ή υπο-προσαρμογής.

Traffic Type



Η καμπύλη **εκπαίδευσης** ξεκινά περίπου από 0.885 και αυξάνεται σταδιακά, φτάνοντας περίπου το 0.91 στο μέγιστο μέγεθος συνόλου. Αντίθετα, η καμπύλη **validation** ξεκινά από χαμηλότερα επίπεδα (περίπου 0.86) και παρουσιάζει μια πιο απότομη αύξηση στην αρχή, συγκλίνοντας σταδιακά προς την καμπύλη εκπαίδευσης, φτάνοντας περίπου το 0.905. Η διαφορά μεταξύ των δύο καμπυλών, αν και μειώνεται με την αύξηση των δεδομένων, υποδηλώνει ότι το μοντέλο θα μπορούσε να επωφεληθεί περαιτέρω από περισσότερα δεδομένα εκπαίδευσης- λογικό για το μικρό σύνολο δειγμάτων που του δώσαμε.

Μετρικές Ταξινόμησης

Το **SVM** μοντέλο απέδωσε καλά στην ταξινόμηση μεταξύ κανονικής και κακόβουλης κυκλοφορίας (**Label**), επιτυγχάνοντας σχετικά υψηλά ποσοστά ακρίβειας και **F1-score** σε όλα τα υποσύνολα δεδομένων (Train, Validation, Test). Από την άλλη πλευρά, στην πιο απαιτητική ταξινόμηση κατά τύπο κυκλοφορίας (**Traffic Type**), το **SVM** εμφανίζει ικανοποιητική αλλά λίγο χαμηλότερη απόδοση, με **F1-score** Test περίπου στο **0.9063**. Αυτό οφείλεται κυρίως στη δυσκολία του μοντέλου να χειριστεί ορθά τις λιγότερο συχνές κατηγορίες, κάτι που αντικατοπτρίζεται και στη μικρή πτώση των επιδόσεων σε σχέση με τις πλειοψηφικές κλάσεις.

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	Sampling	Label	0.944907	0.942064	0.939879	0.940743	0.938040	0.934678
1	Sampling	Traffic Type	0.912431	0.910638	0.906327	0.887092	0.887608	0.878963

Label

```
>>> Training Report
              precision    recall  f1-score   support

      Benign       0.69      0.97      0.80       781
   Malicious       1.00      0.94      0.97      5463

    accuracy                           0.94      6244
   macro avg       0.84      0.95      0.88      6244
weighted avg       0.96      0.94      0.94      6244

>>> Validation Report
              precision    recall  f1-score   support

      Benign       0.68      0.94      0.79       260
   Malicious       0.99      0.94      0.96      1822

    accuracy                           0.94      2082
   macro avg       0.84      0.94      0.88      2082
weighted avg       0.95      0.94      0.94      2082

>>> Test Report
              precision    recall  f1-score   support

      Benign       0.66      0.98      0.79       260
   Malicious       1.00      0.93      0.96      1822

    accuracy                           0.93      2082
   macro avg       0.83      0.95      0.88      2082
weighted avg       0.96      0.93      0.94      2082
```

Traffic Type

>>> Training Report

	precision	recall	f1-score	support
Audio	0.80	0.77	0.79	114
Background	0.09	0.95	0.16	20
Bruteforce	0.58	0.91	0.71	23
DoS	0.99	0.92	0.95	4732
Information Gathering	0.91	0.91	0.91	649
Mirai	0.14	0.81	0.24	59
Text	0.90	0.78	0.84	125
Video	0.82	0.59	0.69	522
accuracy			0.89	6244
macro avg	0.65	0.83	0.66	6244
weighted avg	0.95	0.89	0.91	6244

>>> Validation Report

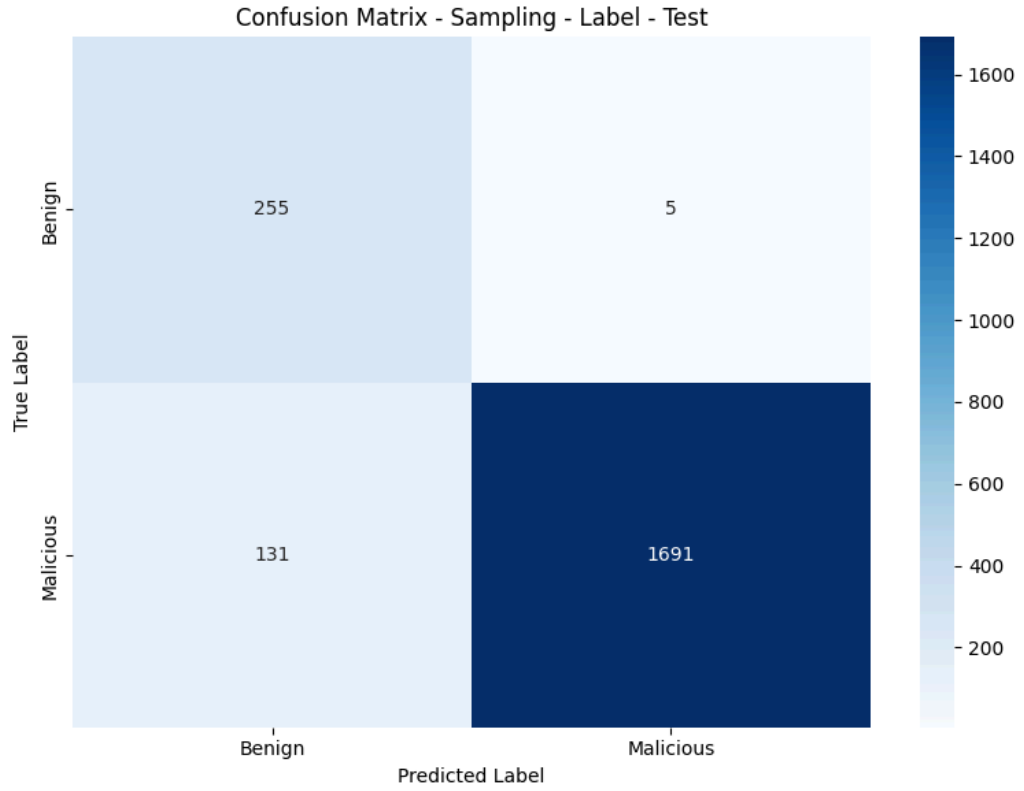
	precision	recall	f1-score	support
Audio	0.80	0.63	0.71	38
Background	0.10	1.00	0.18	6
Bruteforce	0.54	1.00	0.70	7
DoS	0.99	0.92	0.96	1578
Information Gathering	0.91	0.94	0.92	217
Mirai	0.14	0.80	0.24	20
Text	0.80	0.76	0.78	42
Video	0.78	0.59	0.67	174
accuracy			0.89	2082
macro avg	0.63	0.83	0.64	2082
weighted avg	0.95	0.89	0.91	2082

>>> Test Report

	precision	recall	f1-score	support
Audio	0.82	0.71	0.76	38
Background	0.10	1.00	0.19	6
Bruteforce	0.35	0.86	0.50	7
DoS	1.00	0.91	0.95	1578
Information Gathering	0.87	0.94	0.90	217
Mirai	0.10	0.65	0.18	20
Text	0.79	0.74	0.77	42
Video	0.81	0.60	0.69	174
accuracy			0.88	2082
macro avg	0.61	0.80	0.62	2082
weighted avg	0.95	0.88	0.91	2082

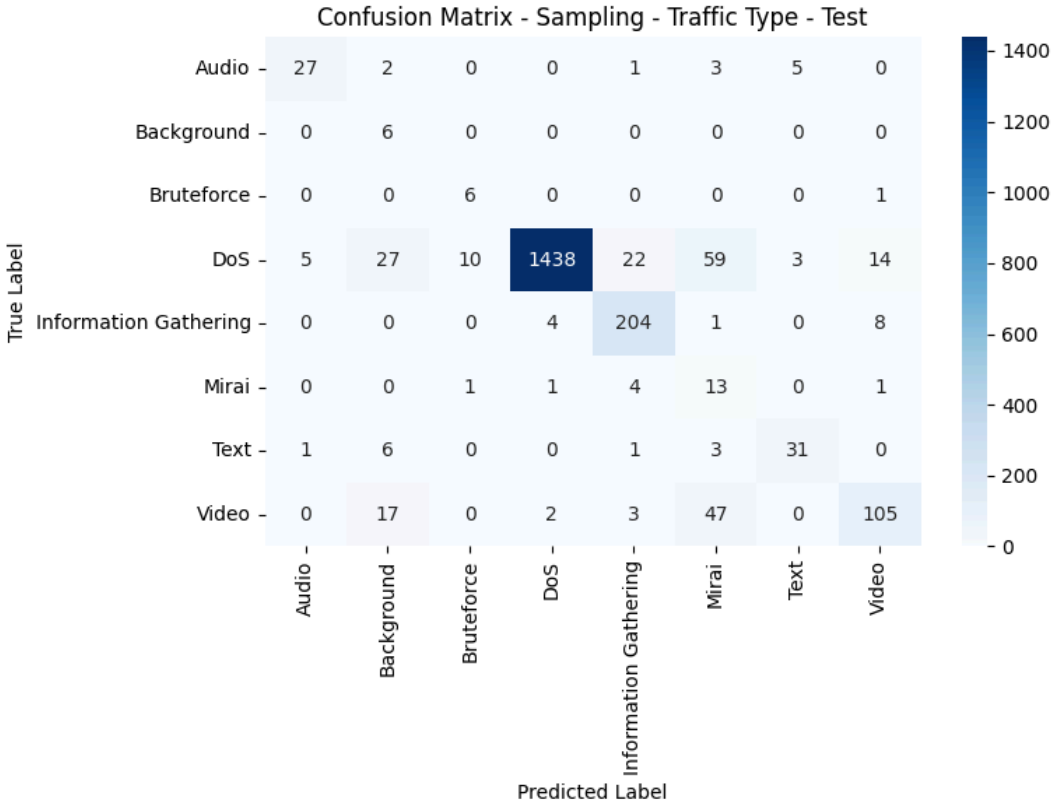
Πίνακας σύγχυσης του μοντέλου MLP

Label



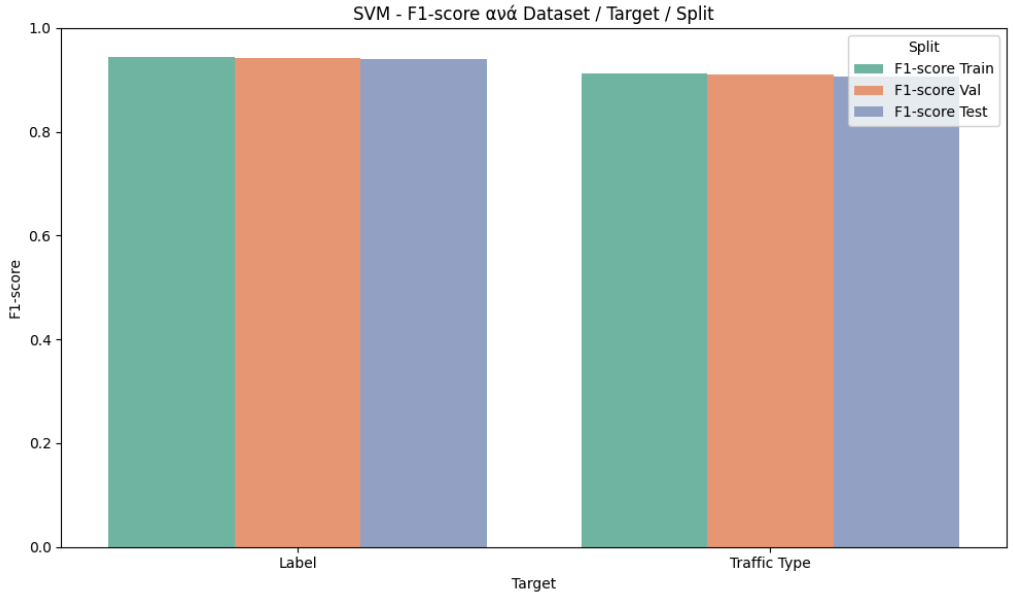
Παρατηρούμε ότι το μοντέλο ταξινομεί με επιτυχία το μεγαλύτερο μέρος των περιπτώσεων. Συγκεκριμένα, 1691 περιπτώσεις "**Malicious**" αναγνωρίστηκαν σωστά (True Positives) και 255 περιπτώσεις "**Benign**" αναγνωρίστηκαν επίσης σωστά (True Negatives). Ωστόσο, υπάρχει ένας σημαντικός αριθμός ψευδώς αρνητικών (False Negatives), όπου 131 περιπτώσεις "**Malicious**" ταξινομήθηκαν λανθασμένα ως "Benign". Οι ψευδώς θετικές προβλέψεις, όπου 5 περιπτώσεις "**Benign**" χαρακτηρίστηκαν ως "Malicious", είναι πολύ λίγες.

Traffic Type



Παρατηρούμε ότι ενώ το μοντέλο επιτυγχάνει υψηλή ακρίβεια σε ορισμένες κατηγορίες, όπως το "**DoS**" (1438 σωστές προβλέψεις), υπάρχουν συγχύσεις σε άλλες. Για παράδειγμα , η κατηγορία "**Video**" εμφανίζει 105 σωστές προβλέψεις, η κατηγορία "**Background**" έχει μόνο 6 σωστές προβλέψεις, ενώ 59 δείγματα "**DoS**" ταξινομούνται λανθασμένα ως "**Mirai**". Αυτά τα ευρήματα υποδηλώνουν ότι το μοντέλο δυσκολεύεται να διακρίνει αποτελεσματικά μεταξύ ορισμένων τύπων κίνησης, λόγω της επίδρασης της δειγματοληψίας και του μικρού αριθμού δειγμάτων.

Αναπαράσταση F1- scores



Το διάγραμμα απεικονίζει τις επιδόσεις του μοντέλου **SVM** σε σχέση με το **F1-score** για τις κλάσεις "**Label**" και "**Traffic Type**", διαχωρισμένες σε σύνολα **εκπαίδευσης** (Train), **επικύρωσης** (Val) και **ελέγχου** (Test). Για την κατηγορία "**Label**", παρατηρείται εξαιρετικά υψηλό **F1-score**, το οποίο κυμαίνεται περίπου στο **0.93-0.94** σε όλα τα σύνολα, με το **Validation set** να εμφανίζει ελαφρώς την καλύτερη επίδοση, υποδεικνύοντας ισχυρή γενίκευση του μοντέλου. Αντίθετα, για την κατηγορία "**Traffic Type**", τα **F1-scores** είναι χαμηλότερα, κινούμενα περίπου από **0.87-0.91**, αν και παραμένουν σε ικανοποιητικά επίπεδα. Η σχετική ομοιομορφία των **F1-scores** μεταξύ των τριών συνόλων και για τις δύο κατηγορίες υποδηλώνει ότι το μοντέλο δεν κάνει **overfitting** και γενικεύει καλά στα καινούρια δεδομένα, παρά τη διαφορά στην απόλυτη τιμή του **F1-score** μεταξύ των δύο στόχων.

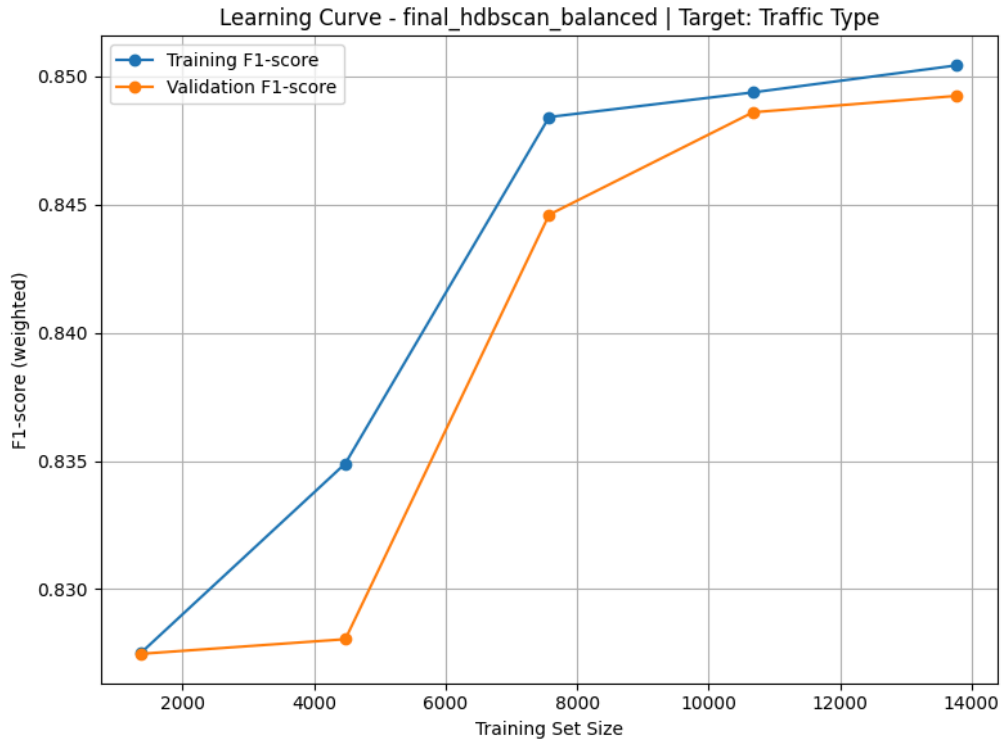
➢ Αποτελέσματα για το σύνολο δεδομένων που παρήγαγε ο HDBSCAN

Καμπύλες μάθησης

Label

Και οι δύο καμπύλες, τόσο της εκπαίδευσης όσο και της επικύρωσης, εμφανίζουν ανοδική τάση, υποδεικνύοντας ότι το μοντέλο βελτιώνεται καθώς αυξάνεται η ποσότητα των δεδομένων εκπαίδευσης. Το **F1-score** εκπαίδευσης είναι ελαφρώς υψηλότερο από το **F1-score** επικύρωσης σε όλο το εύρος, κάτι που είναι αναμενόμενο. Καθώς το μέγεθος του συνόλου εκπαίδευσης πλησιάζει τις 14000 μονάδες, οι δύο καμπύλες συγκλίνουν, με το **F1-score** επικύρωσης να φτάνει σχεδόν το 0.97, γεγονός που υποδηλώνει ότι το μοντέλο έχει μάθει καλά και γενικεύει αποτελεσματικά, χωρίς σημάδια **overfitting ή underfitting.**

Traffic Type



Παρατηρείται ανοδική τάση και στις δύο καμπύλες, με το **F1-score** να βελτιώνεται καθώς διατίθενται περισσότερα δεδομένα εκπαίδευσης. Το **F1-score εκπαίδευσης** είναι σταθερά υψηλότερο από το **F1-score επικύρωσης**, ειδικά σε μικρότερα σύνολα δεδομένων, γεγονός που υποδηλώνει ότι το μοντέλο μπορεί να υπερ-προσαρμόζεται ελαφρώς σε μικρά σύνολα. Ωστόσο, καθώς το μέγεθος του συνόλου εκπαίδευσης αυξάνεται προς τις 14000 μονάδες, οι δύο καμπύλες συγκλίνουν, με το **F1-score** επικύρωσης να φτάνει περίπου το 0.85, δείχνοντας ότι το μοντέλο γενικεύει αρκετά καλά. Υπάρχει περιθώριο βελτίωσης.

Μετρικές Ταξινόμησης

Το **SVM** μοντέλο απέδωσε καλά στην ταξινόμηση μεταξύ κανονικής και κακόβουλης κυκλοφορίας (**Label**), επιτυγχάνοντας υψηλά ποσοστά ακρίβειας και **F1-score** σε όλα τα υποσύνολα δεδομένων (Train, Validation, Test). Από την άλλη πλευρά, στην πιο απαιτητική ταξινόμηση κατά τύπο κυκλοφορίας (**Traffic Type**), το **SVM** εμφανίζει ικανοποιητική αλλά χαμηλότερη απόδοση, με **F1-score** Test περίπου στο **0.83**. Αυτό οφείλεται κυρίως στη δυσκολία του μοντέλου να χειριστεί ορθά τις λιγότερο συχνές κατηγορίες, κάτι που αντικατοπτρίζεται και στη μικρή πτώση των επιδόσεων σε σχέση με τις πλειοψηφικές κλάσεις.

Συνοπτικά Αποτελέσματα:

	Dataset	Target	F1-score Train	F1-score Val	F1-score Test	Accuracy Train	Accuracy Val	Accuracy Test
0	final_hdbscan_balanced	Label	0.970707	0.961813	0.966093	0.970265	0.960976	0.965621
1	final_hdbscan_balanced	Traffic Type	0.851388	0.848920	0.844446	0.840716	0.839024	0.836005

Label

```
>>> Training Report
              precision    recall  f1-score   support

      Benign       0.89      0.97      0.93      2622
   Malicious       0.99      0.97      0.98     10292

    accuracy                           0.97     12914
   macro avg       0.94      0.97      0.96     12914
weighted avg       0.97      0.97      0.97     12914

>>> Validation Report
              precision    recall  f1-score   support

      Benign       0.86      0.97      0.91       874
   Malicious       0.99      0.96      0.98      3431

    accuracy                           0.96      4305
   macro avg       0.93      0.96      0.94      4305
weighted avg       0.96      0.96      0.96      4305

>>> Test Report
              precision    recall  f1-score   support

      Benign       0.89      0.95      0.92       874
   Malicious       0.99      0.97      0.98      3431

    accuracy                           0.97      4305
   macro avg       0.94      0.96      0.95      4305
weighted avg       0.97      0.97      0.97      4305
```

Traffic Type

>>> Training Report

	precision	recall	f1-score	support
Audio	0.95	0.81	0.88	300
Background	0.17	0.98	0.28	300
Bruteforce	0.31	0.94	0.47	54
DoS	1.00	0.93	0.96	7370
Information Gathering	0.95	0.99	0.97	2718
Mirai	0.54	0.93	0.69	150
Text	0.96	0.77	0.85	300
Video	0.69	0.20	0.31	1722
accuracy			0.84	12914
macro avg	0.70	0.82	0.68	12914
weighted avg	0.92	0.84	0.85	12914

>>> Validation Report

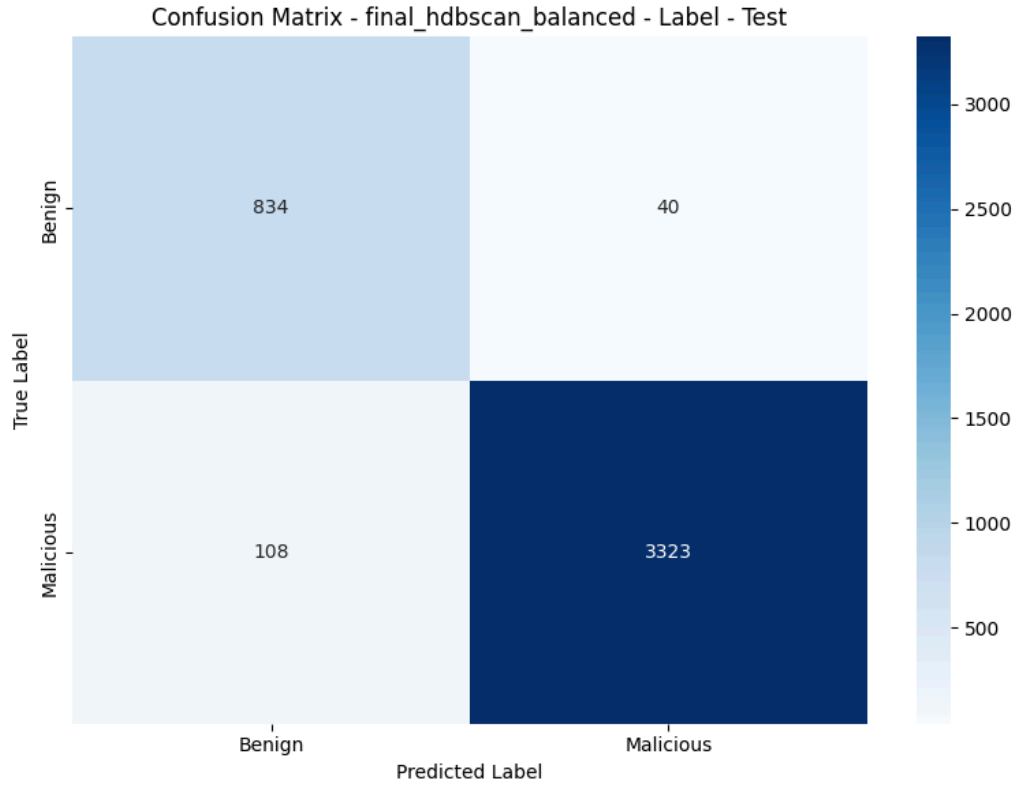
	precision	recall	f1-score	support
Audio	0.91	0.78	0.84	100
Background	0.16	0.95	0.27	100
Bruteforce	0.33	1.00	0.49	18
DoS	0.99	0.93	0.96	2457
Information Gathering	0.94	0.99	0.97	906
Mirai	0.54	0.90	0.68	50
Text	0.95	0.73	0.82	100
Video	0.73	0.20	0.31	574
accuracy			0.84	4305
macro avg	0.69	0.81	0.67	4305
weighted avg	0.92	0.84	0.85	4305

>>> Test Report

	precision	recall	f1-score	support
Audio	0.90	0.78	0.83	100
Background	0.16	0.95	0.27	100
Bruteforce	0.30	0.89	0.45	18
DoS	0.99	0.93	0.96	2457
Information Gathering	0.96	1.00	0.98	906
Mirai	0.60	1.00	0.75	50
Text	0.96	0.74	0.84	100
Video	0.64	0.16	0.26	574
accuracy			0.84	4305
macro avg	0.69	0.81	0.67	4305
weighted avg	0.91	0.84	0.84	4305

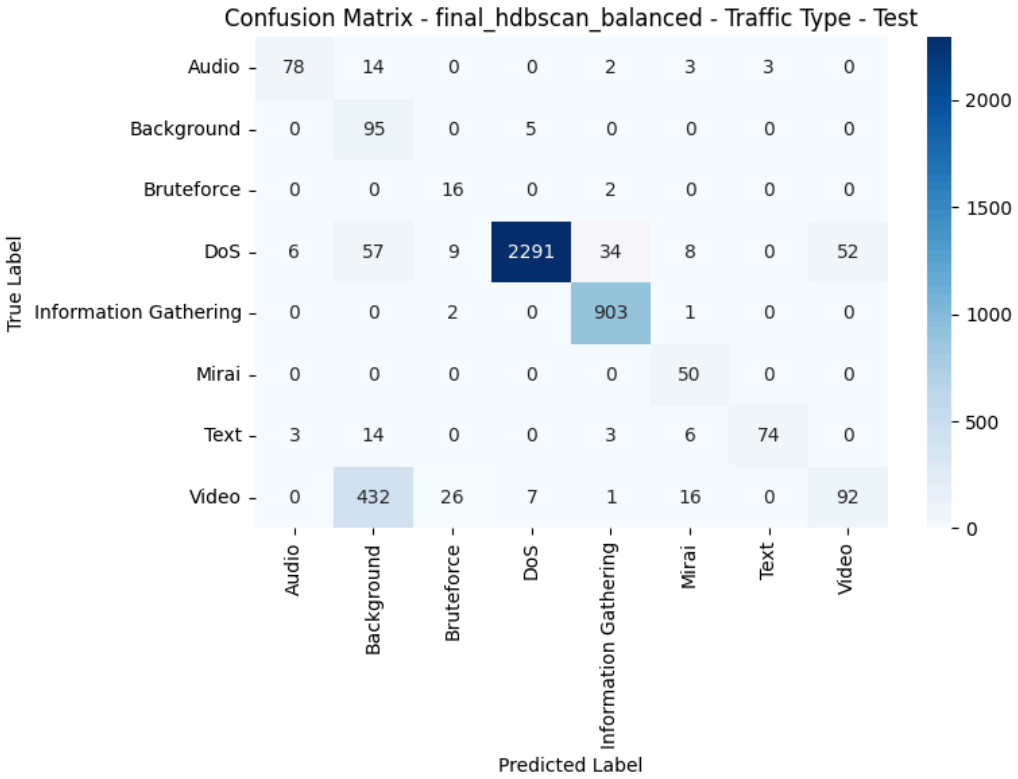
Πίνακας σύγχυσης του μοντέλου MLP

Label



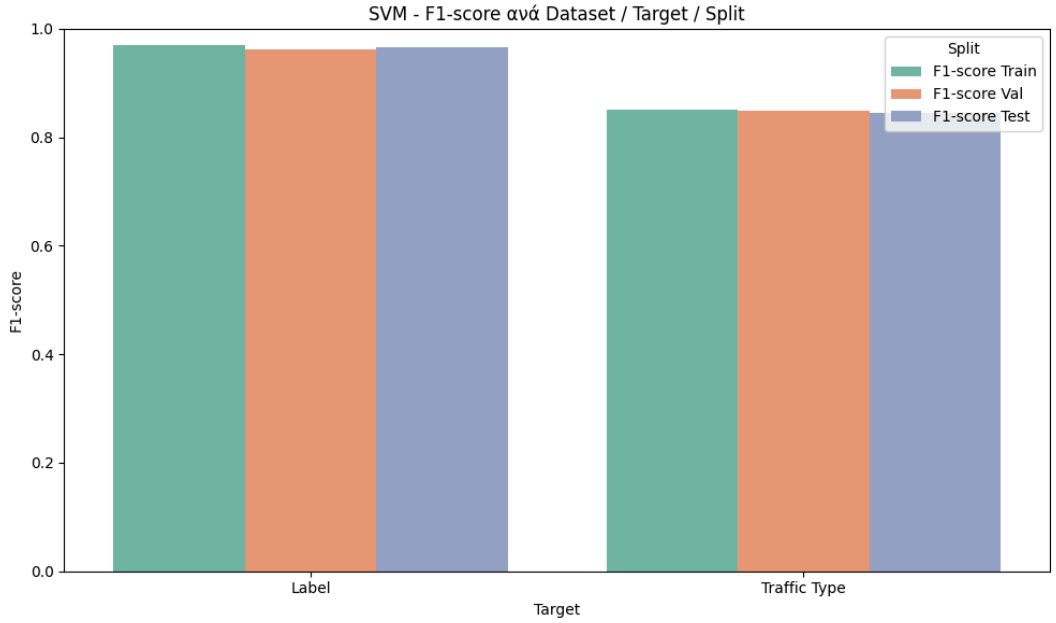
Παρατηρούμε ότι το μοντέλο επιδεικνύει εξαιρετικά υψηλή ακρίβεια στην πρόβλεψη και των δύο κλάσεων. Συγκεκριμένα, 3323 περιπτώσεις "**Malicious**" και 834 περιπτώσεις "**Benign**" ταξινομήθηκαν σωστά. Ο αριθμός των λανθασμένων προβλέψεων είναι μικρός, με 40 περιπτώσεις "**Benign**" να ταξινομούνται εσφαλμένα ως "**Malicious**" (ψευδώς θετικά) και 108 περιπτώσεις "**Malicious**" να ταξινομούνται λανθασμένα ως "**Benign**" (ψευδώς αρνητικά). Τα αποτελέσματα αυτά υποδηλώνουν ένα πολύ αποτελεσματικό μοντέλο στην διάκριση μεταξύ των 2 δραστηριοτήτων.

Traffic Type



Παρατηρούμε υψηλή ακρίβεια σε ορισμένες κατηγορίες, όπως το "**DoS**" (2291 σωστές προβλέψεις) και το "**Information Gathering**" (903 σωστές προβλέψεις), με πολύ λίγες λανθασμένες ταξινομήσεις. Ωστόσο, υπάρχουν ορισμένες κατηγορίες όπου παρατηρείται σύγχυση. Για παράδειγμα, ένα σημαντικό μέρος των περιπτώσεων "**Video**" (432) ταξινομείται λανθασμένα ως "**Background**", και αρκετές περιπτώσεις "**Background**" (95) ταξινομούνται σωστά αλλά και 57 ως "**DoS**" και 14 ως "**Audio**". Επίσης, η κατηγορία "**Audio**" έχει 78 σωστές προβλέψεις αλλά και 14 ως "**Background**". Γενικά, το μοντέλο αποδίδει καλά στις πιο ευδιάκριτες κατηγορίες, αλλά φαίνεται να χρειάζεται βελτίωση στη διάκριση μεταξύ κάποιων, πιο ομοειδών, τύπων κίνησης.

Αναπαράσταση F1- scores



Για τον στόχο "**Label**", το **F1-score** είναι κυμαίνεται υψηλά και παραμένει σταθερό με μικρές διαφορές σε όλα τα σύνολα, υποδεικνύοντας καλή γενίκευση και έλλειψη υπερ-προσαρμογής. Όσον αφορά τον στόχο "**Traffic Type**", το **F1-score** παραμένει σε καλά επίπεδα (περίπου 0.85), αν και είναι χαμηλότερο από τον στόχο "**Label**". Η σταθερότητα του **F1-score** σε όλα τα **splits** για τον "**Traffic Type**" δείχνει επίσης καλή γενίκευση, παρόλο που η συνολική απόδοση είναι λίγο χαμηλότερη σε σχέση με την ταξινόμηση "**Label**".

Συμπέρασμα

Τα καλύτερα αποτελέσματα συνολικά επιτεύχθηκαν από το μοντέλο **MLP - Neural Network**, με κορυφαία απόδοση στο σύνολο δεδομένων που δημιουργήθηκε μέσω δειγματοληψίας (**Stratified Sampling**). Σε αυτό το dataset, το **MLP** πέτυχε σχεδόν τέλεια αποτελέσματα τόσο στην ταξινόμηση της κυκλοφορίας ως **Benign/Malicious** (**Label**) όσο και στους τύπους κίνησης (**Traffic Type**), με **F1-scores** που αγγίζουν το 1.0 και 0.99 αντίστοιχα, χωρίς ενδείξεις **overfitting**. Πολύ καλά αποτελέσματα σημειώθηκαν και στο **HDBSCAN**, ενώ στο **MiniBatchKMeans** η απόδοση του **MLP** ήταν ελαφρώς χαμηλότερη, κυρίως λόγω μικρών κατηγοριών που δεν αναγνωρίστηκαν επαρκώς.

Από την άλλη πλευρά, το **SVM** απέδωσε επίσης ιδιαίτερα καλά στην κατηγοριοποίηση Label, επιτυγχάνοντας F1-score έως και 0.945, ειδικά στο δειγματοληπτικό σύνολο, αποδεικνύοντας ότι μπορεί να διαχειριστεί αποτελεσματικά το πρόβλημα δυαδικής ταξινόμησης. Ωστόσο, στην ταξινόμηση του **Traffic Type**, η απόδοσή του ήταν αισθητά χαμηλότερη, με **F1-scores** γύρω στο 0.80, και εμφανή δυσκολία στη διάκριση μεταξύ παρόμοιων τύπων κυκλοφορίας όπως "**Video**", "**Text**", "**Audio**" και "**Background**". Επιπλέον, παρουσίασε περισσότερες εσφαλμένες προβλέψεις (false positives/negatives) σε σχέση με το **MLP**. Συμπερασματικά, το **MLP** υπερέχει καθαρά έναντι του **SVM** στην πολυκατηγορική πρόβλεψη και γενικά στα περισσότερα σύνολα, με τη βέλτιστη συνολική επίδοση να επιτυγχάνεται στο

δειγματοληπτικό dataset. Το **SVM** παραμένει αξιόπιστο και αποδοτικό εργαλείο για το **Label**, αλλά λιγότερο κατάλληλο για σύνθετα προβλήματα ταξινόμησης όπως το **Traffic Type**.