# ΥΛΟΠΟΙΗΤΙΚΟ PROJECT ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ 2024 – 2025

ΚΩΝΣΤΑΝΤΙΝΟΣ ΒΑΣΙΛΑΔΙΩΤΗΣ, ΑΜ: 1088094

ΑΓΓΕΛΟΣ ΚΟΝΤΑΛΗΣ, ΑΜ: 1095483

Σε αυτήν την Εργασία χρησιμοποιήσαμε ως περιβάλλον υλοποίησης το Anaconda Navigator και κατά προέκταση το Jupyter Notebook για την ανάλυση των δεδομένων και την εξαγωγή των συμπερασμάτων μας.

Αρχικά οι βιβλιοθήκες που χρησιμοποιήσαμε είναι οι εξής:

Pandas, numpy, matplotlib, seaborn, tensorflow, keras και sklearn.

**Προσέγγιση Dataset:**

Στο συγκεκριμένο Project το dataset ήταν αρκετά μεγάλο σε χωρητικότητα (5gb) κάτι που μας δημιούργησε προβλήματα κατά το άνοιγμά του με την Pandas. Για να λύσουμε αυτό το πρόβλημα εφαρμόσαμε μια υποδειγματοληψία στην οποία κρατήσαμε ένα ποσοστό από κάποια δείγματα που εμφανίζονται υπερβολικά πολλές φορές, ενώ ταυτόχρονα κρατήσαμε όλα τα σπάνια. Με αυτό τον τρόπο το αρχείο csv άνοιγε επιτυχώς διατηρώντας ταυτόχρονα τις αναλογίες των δεδομένων του.

**Ερώτημα 1:**

Ξεκινήσαμε την ανάλυση του dataset πραγματοποιώντας κάποιες βασικές εντολές όπως df.head() για να δούμε το περιεχόμενο κάποιων στηλών, df.info() για να δούμε το type των χαρακτηριστικών μας (καθώς στην πορεία οι στήλες object θα χρειαστούν μετατροπή σε numerical) και df.describe() για κάποιες επιπλέον πληροφορίες.

Έπειτα γνωρίζοντας ότι ο στόχος μας είναι η πρόβλεψη της τιμής του column “Label”, έπρεπε να πραγματοποιήσουμε ανάλυση των δεδομένων με βάση αυτή την στήλη, να δούμε τις σχέσεις της με τις υπόλοιπες, ποιες στήλες την επηρεάζουν περισσότερο και ποιες όχι, τα στατιστικά των τιμών της κλπ...

```
[5]: df["Label"].value_counts()

[5]: Label
     Malicious    60166
     Benign        1301
     Name: count, dtype: int64
```

Για παράδειγμα με χρήση της εντολής .value_counts() βλέπουμε ότι στο αρχείο μας τα κακόβουλα Label υπερτερούν κατά πολύ των καλόβουλων και στην συνεχεία πρέπει να βρούμε τι καθορίζει αυτή την διαφορά.

| | Label | Traffic Type | Counts |
|---|---|---|---|
| 0 | Benign | Audio | 190 |
| 1 | Benign | Background | 32 |
| 2 | Benign | Text | 209 |
| 3 | Benign | Video | 870 |
| 4 | Malicious | Bruteforce | 13100 |
| 5 | Malicious | DoS | 37523 |
| 6 | Malicious | Information Gathering | 5167 |
| 7 | Malicious | Mirai | 4376 |

Αναλύσαμε τις συσχετίσεις διάφορων στηλών με την “Label” βλέποντας κάθε φορά την επιρροή που ασκεί μια άλλη στήλη. Αυτό στο συγκεκριμένο παράδειγμα φαίνεται από τα Counts που έχουν μεγάλες διαφορές στις τιμές τους.

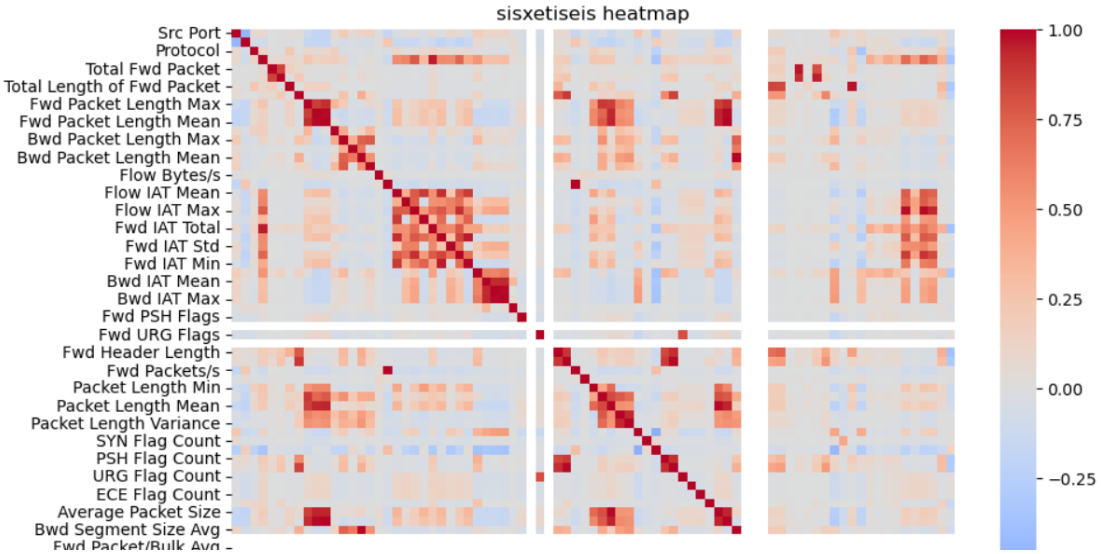
[20]:

| Label | Bwd Packet Length Max | Bwd Packet Length Min | Fwd Packet Length Max | Fwd Packet Length Min |
|---|---|---|---|---|
| Benign | 309.688701 | 1.490392 | 361.805534 | 247.723290 |
| Malicious | 50.844264 | 0.273460 | 265.392730 | 223.011701 |

Άλλες τεχνικές που χρησιμοποιήσαμε ήταν για παράδειγμα πάλι με Target Column = “Label” η ανάλυση των μέσων όρων κάποιων στηλών ανάλογα με την τιμή Malicious η Benign. Στην συγκεκριμένη τεχνική όπου είδαμε μεγάλες

διαφορές στις τιμές θεωρήσαμε την συγκεκριμένη στήλη ως σχετική καθώς επηρεάζει.

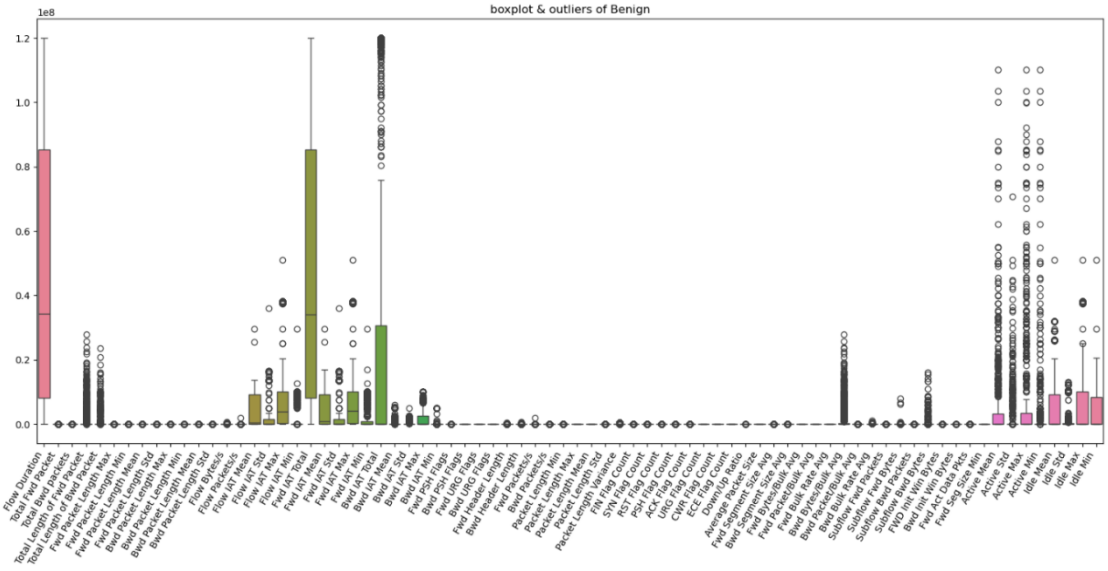
Στην συνέχεια διάφορα διαγράμματα όπως το heatmap βοήθησαν στην ανάλυση του dataset, διακρίνοντας τις συσχετίσεις μεταξύ των στηλών όπως για παράδειγμα πότε όταν αυξάνεται η τιμή μιας αυξάνεται η τιμή μίας άλλης.



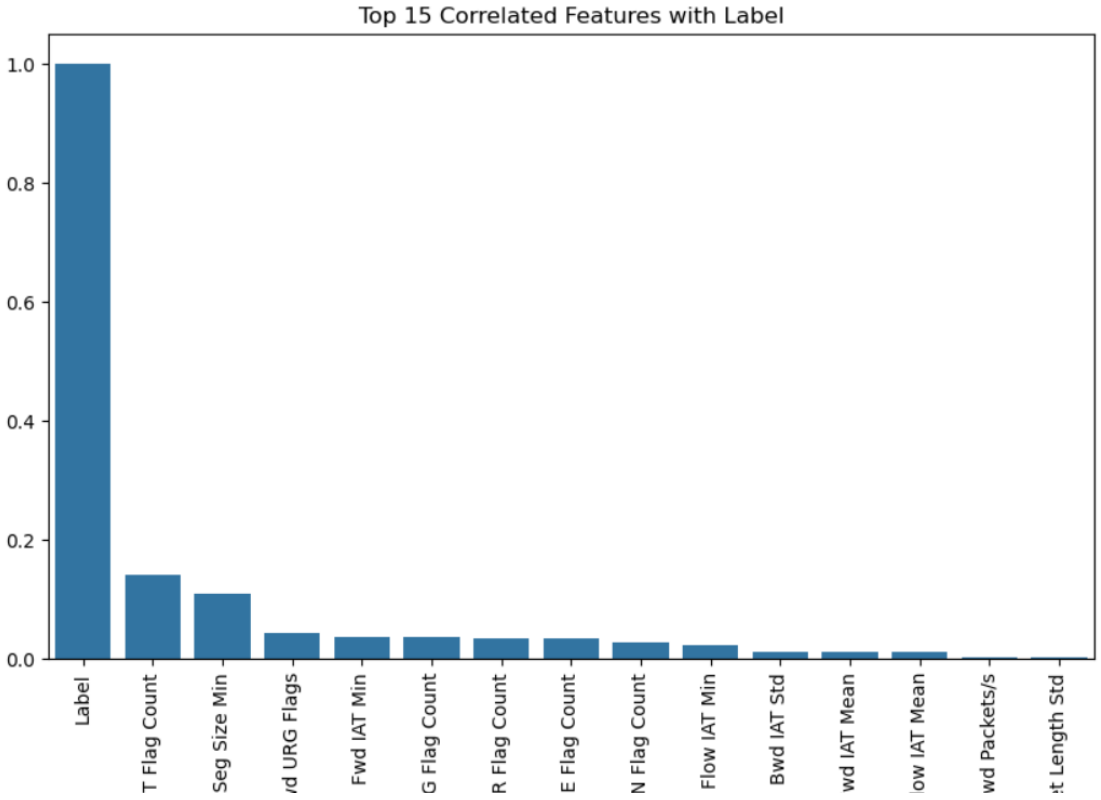

**Ερώτημα 2:**

Πολλές από τις προηγούμενες εντολές μας βοήθησαν και σε αυτό το ερώτημα καθώς ο στόχος είναι να μειώσουμε το σύνολο δεδομένων κρατώντας όσο το δυνατόν περισσότερη και χρησιμότερη πληροφορία.

Όπως και προηγουμένως με στόχο το target column μας η ανάλυση συνεχίστηκε με τεχνικές όπως η εύρεση και απομάκρυνση των outliers.




Εδώ ασπούμε μπορούμε να διακρίνουμε κάποια columns που με τις ακραίες τιμές τους επηρεάζουν τις γραμμές με Label = “Benign”. Τέτοιες στήλες κάλο είναι να μην χρησιμοποιούνται καθώς μπορεί να συμβάλλουν αρνητικά στην εκπαίδευση του μοντέλου μας.




Εδώ για παράδειγμα αναζητήσαμε τις 15 πιο σχετικές στήλες με την στήλη Label.

Μετά από όλη την ανάλυση των ερωτημάτων 1 & 2 κατασκευάσαμε ένα καινούργιο df_new το οποίο βάση των αποτελεσμάτων μας περιέχει μόνο τις πιο σχετικές στήλες και κατά προέκταση είναι πολύ πιο εύκολα διαχειρίσιμο πάνω στο πρόβλημά μας.

```
features = ['Label','RST Flag Count','Fwd Seg Size Min','Fwd URG Flags','URG Flag Count',
            'ECE Flag Count','CWR Flag Count','Bwd Init Win Bytes','Fwd IAT Min',
            'FIN Flag Count','SYN Flag Count','ACK Flag Count','PSH Flag Count',
             'Bwd Packet Length Max','Bwd Packet Length Min','Fwd Packet Length Max',
             'Fwd Packet Length Min','Packet Length Mean','Packet Length Std','Traffic Type',
            'Traffic Subtype']
df_new = df[features]
```

Έπειτα αφού μετατρέψαμε όλες τις object columns σε numerical ελαττώσαμε τις γραμμές του συνόλου δεδομένων χρησιμοποιώντας 3 τρόπους.

**Α)** Τυχαία δειγματοληψία και κατασκευή ενός νέου μικρότερου συνόλου δεδομένων.

**Β)** Clustering με χρήση του Birch αλγορίθμου και κατασκευή ενός νέου μικρότερου συνόλου δεδομένων.

**Γ)** Clustering με χρήση του Dbscan αλγορίθμου και κατασκευή ενός νέου μικρότερου συνόλου δεδομένων.

Γιατί επιλέξαμε αυτές τις δύο τεχνικές Clustering;

**Birch:**

Ο αλγόριθμος Birch είναι ιδανικός για μεγάλα Datasets καθώς αντί να φορτώνει όλα τα δεδομένα στην μνήμη τα συμπυκνώνει σε ένα clustering tree. Έτσι κρατά μια «περίληψη» των δεδομένων κάτι που τον καθιστά πολύ αποδοτικό σε χρόνο και μνήμη. Επίσης έχει χαμηλή πολυπλοκότητα και μπορεί να δουλέψει και με και χωρίς προκαθορισμένο αριθμό clusters.

**Dbscan:**

Αντίστοιχα και αυτός δεν απαιτεί να γνωρίζουμε πόσα clusters υπάρχουν. Ο αλγόριθμος αυτός λειτουργεί ανακαλύπτοντας περιοχές υψηλής πυκνότητας και σχηματίζει ένα cluster για κάθε μια από αυτές. Κάτι επιπλέον πολύ σημαντικό για την επιλογή αυτού του αλγορίθμου ήταν ότι είναι ιδανικός για πραγματικά δεδομένα καθώς είναι ανθεκτικός σε outliers και θόρυβο. Αυτό σημαίνει ότι σημεία που δεν ανήκουν σε κάποια πυκνή περιοχή και δεν έχουν κάποιο cluster τα θεωρεί ως θόρυβο.

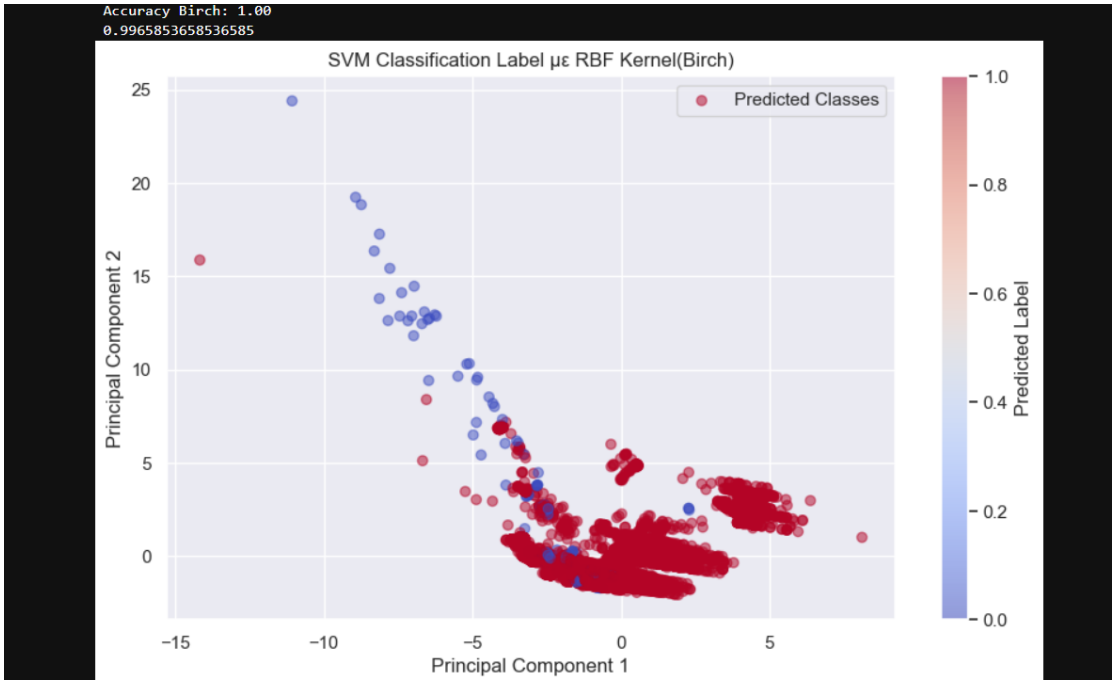
Έτσι στο τέλος του ερωτήματος 2 είχαμε δημιουργήσει 3 νέα σύνολα δεδομένων (df_sampled, df_dbscan, df_birch) όλα πιο μικρά και συμπιεσμένα.

**Ερώτημα 3:**

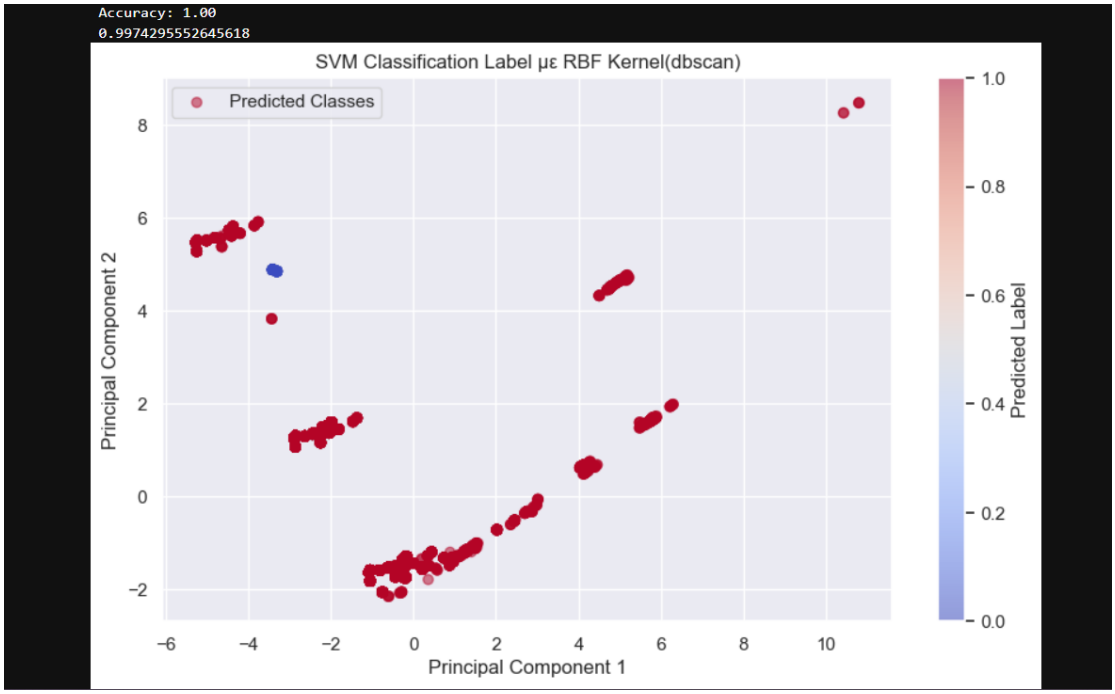
**Classifier SVM:**

Για κάθε ένα από τα προηγούμενα 3 συμπιεσμένα σύνολα δεδομένων χωρίσαμε τα δεδομένα σε y/x όπου το ένα σύνολο αφορά το target column και το άλλο τα υπόλοιπα columns, εφαρμόσαμε μία κανονικοποίηση των δεδομένων, κάναμε διαχωρισμό σε train & test και εκπαιδεύσαμε το μοντέλο SVM. Παρακάτω παραθέτουμε τα αποτελέσματα του SVM για τα 3 σύνολα δεδομένων καθώς και τις μετρικές αξιολόγησης του καθενός.

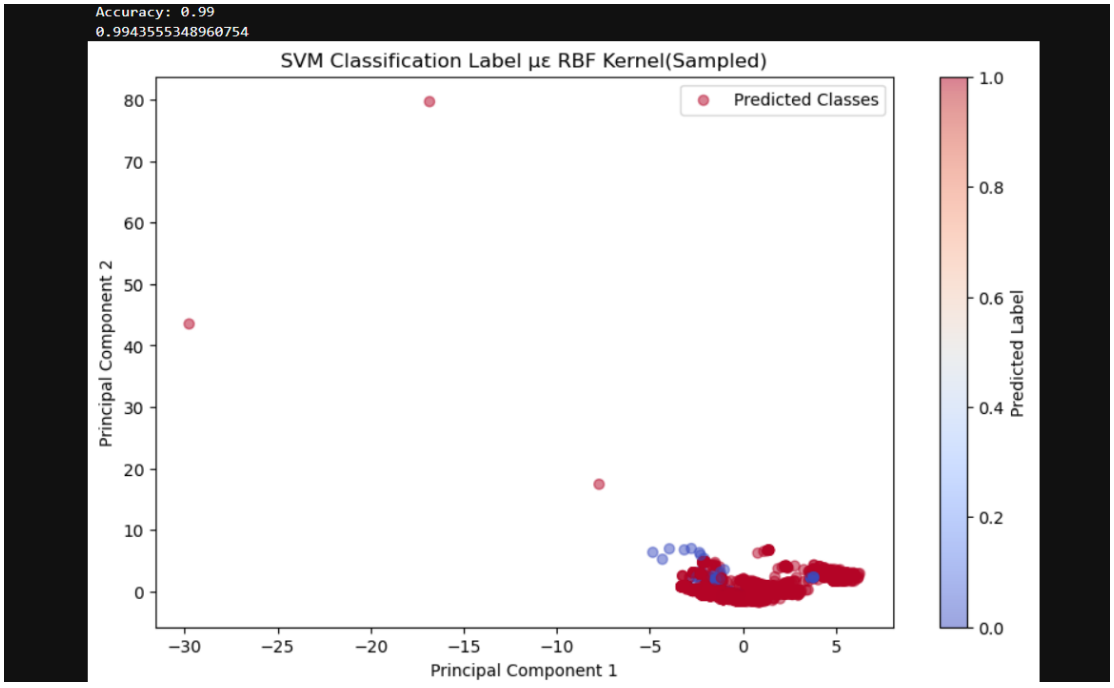
SVM-BIRCH:


Accuracy Birch: 1.00
0.9965853658536585
SVM Classification Label με RBF Kernel(Birch)
Predicted Classes
Principal Component 2
Principal Component 1
Predicted Label
25
20
15
10
5
0
−15
−10
−5
0
5
1.0
0.8
0.6
0.4
0.2
0.0

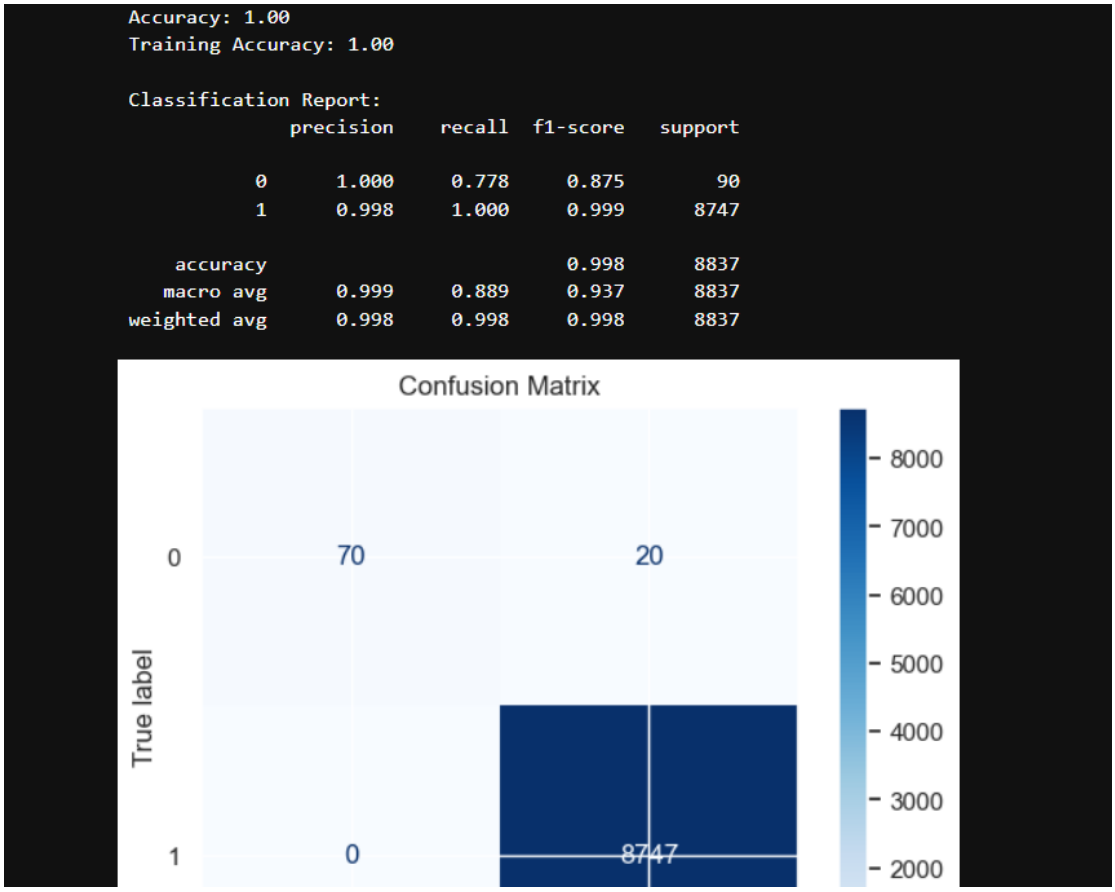
SVM-DBSCAN:




SVM-SAMPLED:



Μετρικές για παράδειγμα DBSCAN:



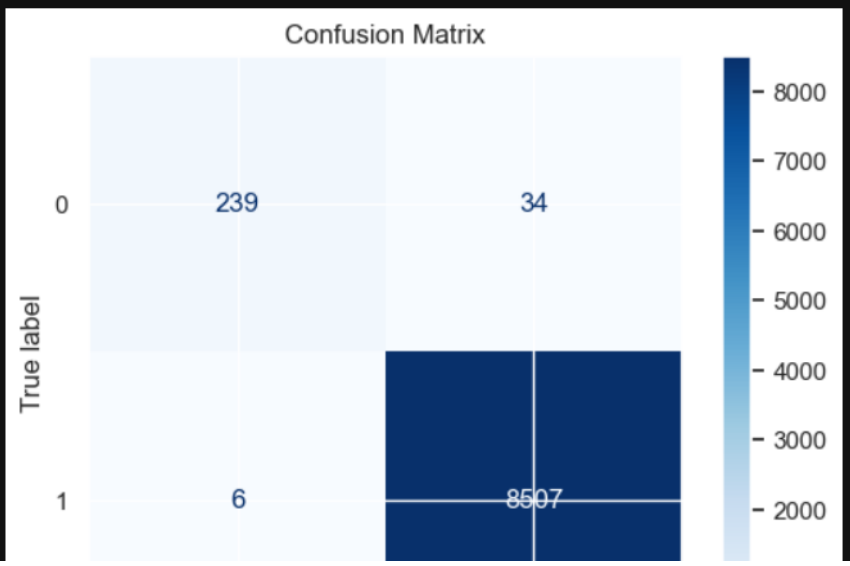

Βλέπουμε accuracy = 1.0 δηλαδή το μοντέλο προβλέπει τα labels στο test set με πολύ μεγάλη ακρίβεια, αλλά επειδή το dataset μας είναι unbalanced σε τιμές malicious/benign δεν αρκεί αυτή η μετρική.

Στο Confusion Matrix βλέπουμε 8747 TP δηλαδή σωστά προβλεπόμενα malicious, 70 TN δηλαδή σωστά προβλεπόμενα benigns, 20 FP που προβλέφθηκαν ως malicious ενώ ήταν benign και 0 FN δηλαδή κανένα malicious δεν προβλέφθηκε λάθος.

Στο Classification Report βλέπουμε για παράδειγμα στην κατηγορία label = 0 (benign), precision= 1.000 δηλαδή όσα προέβλεψε ως 0 ήταν όλα σωστά, Recall = 0.778 δηλαδή από τα πραγματικά 0 βρήκε το 77,8% και f1-score = 0.875 είναι ο συνδιασμός των 2 προηγούμενων μετρικών. Αντίστοιχα και για την κατηγορία labell = 1 (Malicious).

Μετρικές για παράδειγμα BIRCH:

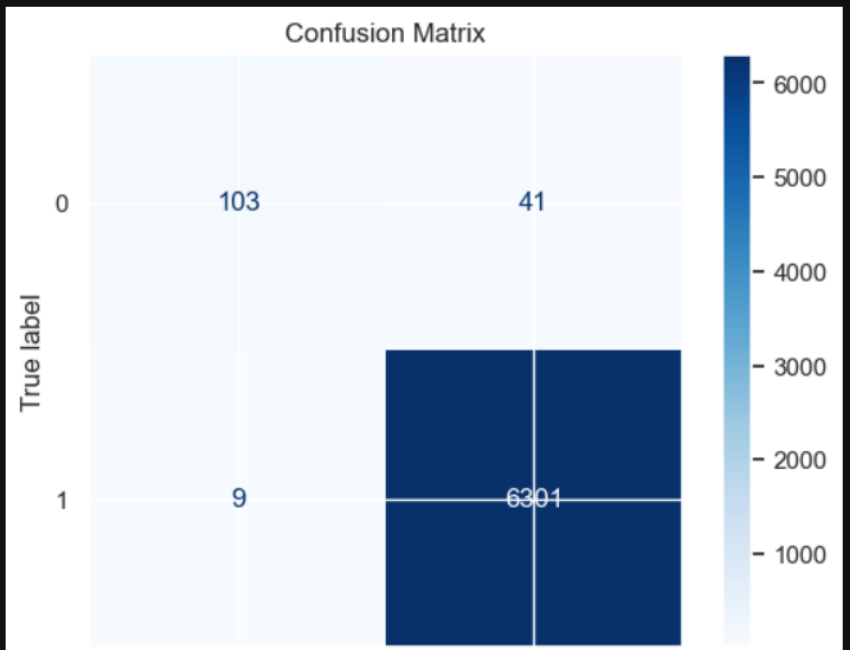
```
Accuracy Birch: 1.00
0.9965853658536585

Classification Report:
              precision    recall  f1-score   support

           0      0.976     0.875     0.923       273
           1      0.996     0.999     0.998      8513

    accuracy                          0.995      8786
   macro avg      0.986     0.937     0.960      8786
weighted avg      0.995     0.995     0.995      8786
```



Μετρικές για παράδειγμα SAMPLED:

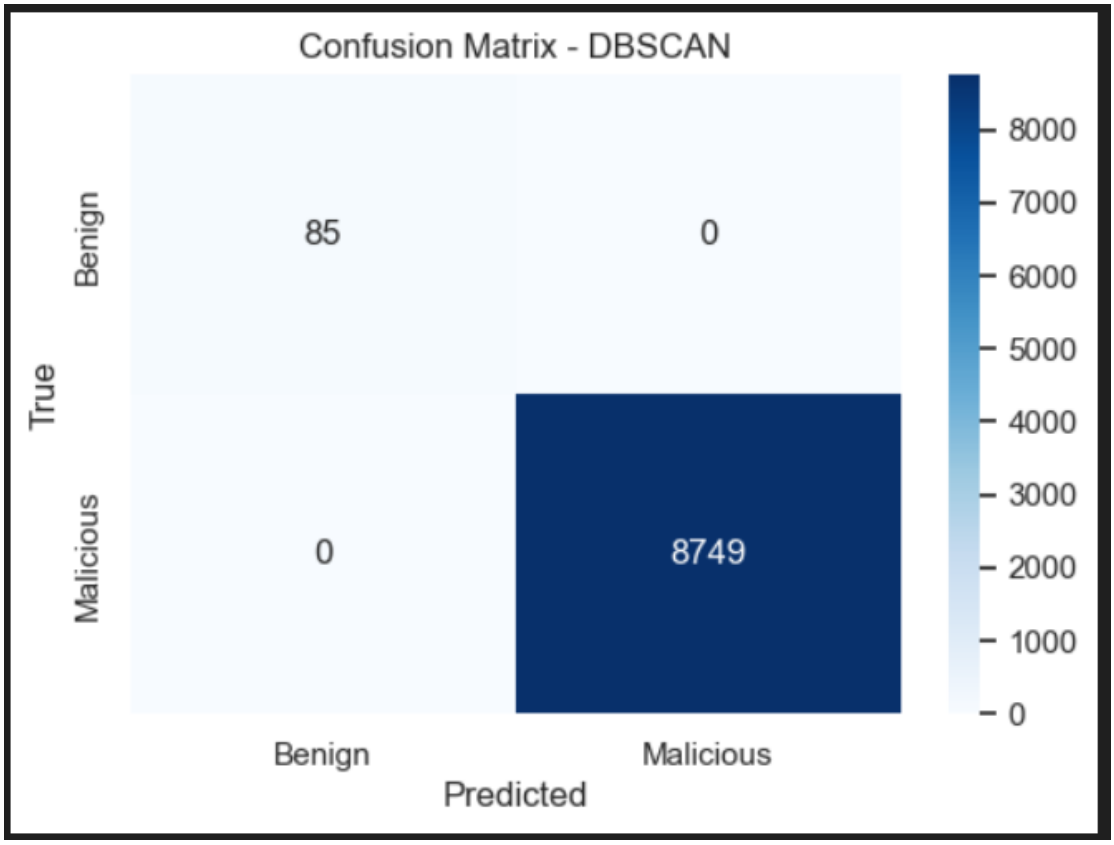
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.920 | 0.715 | 0.805 | 144 |
| 1 | 0.994 | 0.999 | 0.996 | 6310 |
| accuracy | | | 0.992 | 6454 |
| macro avg | 0.957 | 0.857 | 0.900 | 6454 |
| weighted avg | 0.992 | 0.992 | 0.992 | 6454 |

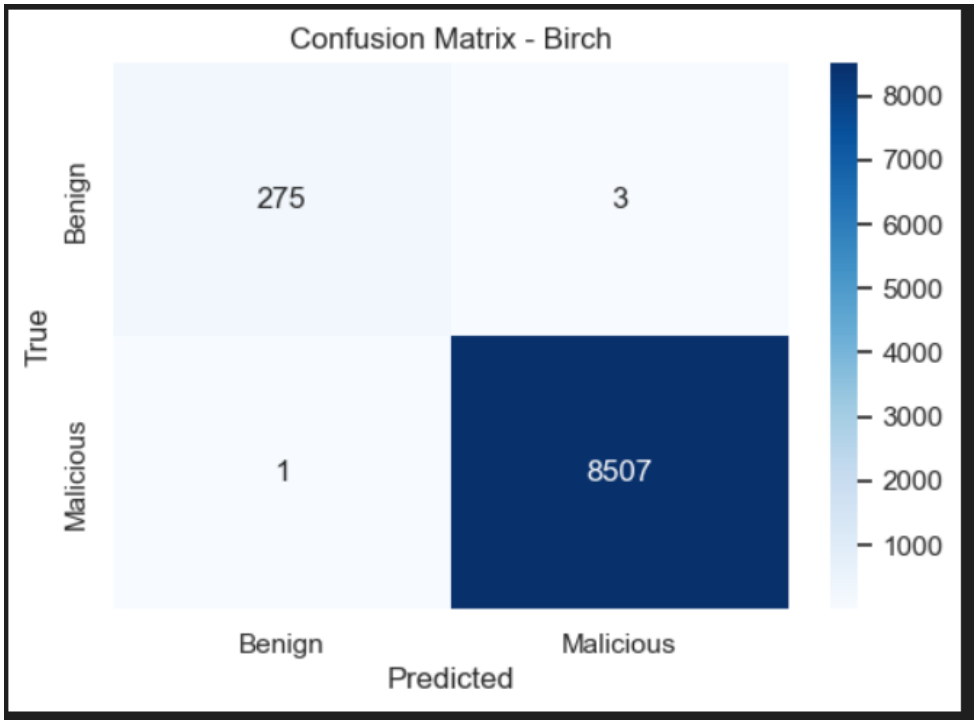



Γενικά με το μοντέλο SVM είδαμε πολύ καλά αποτελέσματα με μικρές διαφορές στις μετρικές ωστόσο τα σύνολα δεδομένων df_dbscan και df_birch ξεχώρισαν με το dbscan να κάνει τα λιγότερα λάθη.

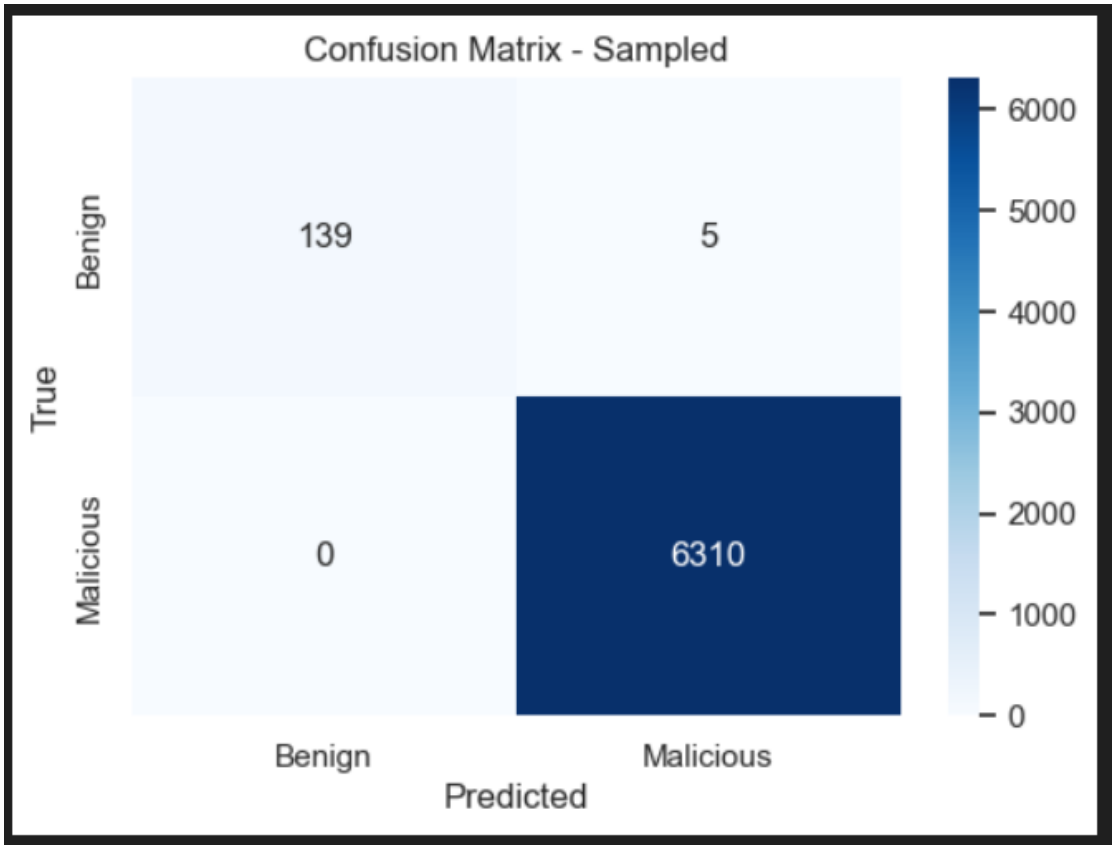
Αντίστοιχα τα αποτελέσματα του Νευρωνικού:




Ο Dbscan έχει προβλέψει 8749 Malicious σωστά, 85 Benign σωστά και 0 λάθος προβλέψεις.



Ο Birch έχει 8507 Malicious σωστά προβλεπόμενα, 275 Benign και 3 και 1 λάθη αντίστοιχα.




Τέλος το Sampled έχει 6310 σωστά Malicious, 139 Benign σωστά και 5 λάθη που προβλέφθηκαν ως Malicious ενώ ήταν Benign.

Αντίστοιχα λοιπόν και το μοντέλο του νευρωνικού είχε πολύ υψηλές αποδόσεις με πάλι το sampled να έχει τα λίγο περισσότερα λάθη ενώ οι αλγόριθμοι Dbscan & Birch να ξεχωρίζουν.