Network Slicing In 6G Networks

Βασίλειος Μάριος Κουρτάκης
ΑΜ:1090061 email: up1090061@ac.upatras.gr

Δημήτριος Στασινός
ΑΜ:1084643 email: up1084643@ac.upatras.gr

Περιεχόμενα

Network Slicing

Το Network Slicing αφορά τον διαχωρισμό ενός δικτύου σε πολλαπλά λογικά δίκτυα τα οποία συνυπάρχουν μέσα σε μια κοινόχρηστη φυσική υποδομή. Κάθε network slice είναι ανεξάρτητο και παρέχει ένα απομονωμένο end-to-end διαδίκτυο με σκοπό την εξυπηρέτηση μια συγκεκριμένης λειτουργίας ή εφαρμογής. Ταυτόχρονα, κάθε slice ορίζει την τοπολογία, τις απαιτήσεις SLA, την αξιοπιστία και το επίπεδο ασφάλειάς του. Υπάρχουν τρεις κατηγορίες slice:

- Enhanced Mobile Broadband (eMBB): Στοχεύει στη ταχύτητα και χωρητικότητα του δικτύου με χαμηλή καθυστέρηση (Video Streaming, Virtual Reality etc.).
- Ultra Reliable Low Latency Communications (URLLC): Εξασφαλίζει την μεταφορά δεδομένων με ελάχιστη καθυστέρηση και μεγάλη αξιοπιστία για υπηρεσίες που η επικοινωνία μέσω του δικτύου είναι κρίσιμη (autonomous vehicles, augmented and virtual reality (AR/VR)).
- Massive or Critical Machine Type Communications (mMTC or CMTC): Στοχεύει στην ευρεία χωρική κάλυψη πολλαπλών συσκευών σε κοντινή απόσταση μεταξύ τους (smart cities, smart health care).

Χαρακτηριστικά των Slices

Isolation (Απομόνωση)

Κάθε slice λειτουργεί ανεξάρτητα από τα υπόλοιπα, εξασφαλίζοντας αποκλειστική χρήση πόρων και αυξημένο επίπεδο ασφάλειας.

Προσαρμογή (Customization)

Τα slices μπορούν να ρυθμιστούν ανάλογα με τις ιδιαίτερες απαιτήσεις διαφορετικών εφαρμογών ή τομέων της βιομηχανίας.

Κατανομή Πόρων (Resource Allocation)

Παρέχεται αποδοτική διαχείριση κρίσιμων πόρων, όπως εύρος ζώνης, υπολογιστική ισχύς και αποθηκευτικός χώρος.

Σημασία ύπαρξης του Network Slicing

Σκοπός ύπαρξης του Network Slicing στα 5G και στα ανερχόμενα 6G δίκτυα είναι κατά κύριο λόγο η δυνατότητα παροχής ευέλικτων και δυναμικών δικτύων στις συνεχώς αυξανόμενες συσκευές και υπηρεσίες που απαιτούν όλο και περισσότερο μεγαλύτερες ταχύτητες και αξιοπιστία. Στοχεύουν, επιπροσθέτως, στο να ανταποκριθούν στις διαφοροποιημένες απαιτήσεις των πελατών για

δυνατότητες δικτύου με μία μόνο φυσική υποδομή μειώνοντας και το κόστος των αναγκαίων υποδομών που χρειάζεται ένας πάροχος.

Machine Learning: Η αναγκαιότητά του στο Network Slicing

Με τη χρήση μοντέλων μηχανικής εκμάθησης, παρέχεται η δυνατότητα πρόβλεψης του slice στο οποίο ανήκει η υπηρεσία καθώς και στη βελτίωση του εύρους ζώνης, ανάλογα τις απαιτήσεις του δικτύου. Έτσι, γίνεται βέλτιστη πρόβλεψη απαιτήσεων (π.χ video: ανήκει στο slice eMBB) και ταυτόχρονα ελαχιστοποίηση των καθυστερήσεων μέσω της αυτόματης λήψης αποφάσεων και μείωση χειροκίνητων διαδικασιών. Σαν αποτέλεσμα, βελτιστοποιείται η ποιότητα της υπηρεσίας καθώς και του χρήστη (QoS/QoE).

Δεδομένα του dataset

Πέρα από τις πρώτες τέσσερις κατηγορίες, οι υπόλοιπες έχουν τις τιμές 0 και 1 ανάλογα με τον αν ανήκει στην συγκεκριμένη κατηγορία ή όχι.

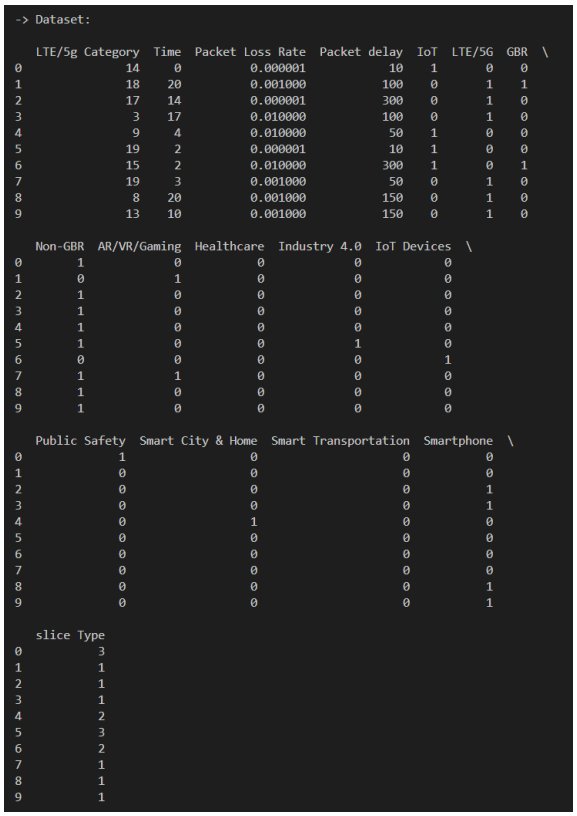
- 0: Δεν ανήκει στην κατηγορία.
- 1: Ανήκει στην κατηγορία.

Επεξήγηση Label

- LTE/5g Category: Κατηγορία LTE/5G
- Time: Ώρα ημέρας (0 έως 23)
- Packet Loss Rate: Ποσοστό απώλειας πακέτων (Νούμερο πακέτων που δε παραλήφθηκαν διαιρεμένο με το συνολικό πλήθος πακέτων που στάλθηκαν)
- Packet Delay: Χρόνος λήψης πακέτου
- IoT: Αν πρόκειται για IoT συσκευή (1) ή όχι (0)
- LTE/5G: Αν πρόκειται για LTE/5G συσκευή (1) ή όχι
- GBR (Guaranteed Bit Rate): Εξασφαλισμένο bit rate από τον πάροχο, 1 εάν παρέχεται και 0 αν δεν παρέχεται
- Non-GBR: Έχει τιμή 1 αν δεν παρέχεται εξασφαλισμένο bit rate και 0 αν παρέχεται
- AR/VR/Gaming: Αν πρόκειται συσκευή AR, VR ή Gaming (1 ή 0)
- Healthcare: Αν πρόκειται για συσκευή που χρησιμοποιείται σε ιατροφαρμακευτική περίθαλψη (1 ή 0)
- Industry 4.0: Αν χρησιμοποιείται σε ψηφιακές εταιρίες (1 ή 0)
- IoT Devices: Αν πρόκειται για IoT (Internet of Things) συσκευής (1 ή 0)
- Public Safety: Αν χρησιμοποιείται για λόγους ασφάλειας και την δημόσια υγεία (1 ή 0)

- Smart City & Home: Αν χρησιμοποιείται σε έξυπνες οικιακές συσκευές (1 ή 0)
- Smart Transportation: Αν χρησιμοποιείται στα MMM (1 ή 0)
- Smartphone: Αν χρησιμοποιείται για δίκτυα κινητής τηλεφωνίας (1 ή 0)
- Slice Type: Το slice στο οποίο ανήκει η συσκευή

Μέσα από το μενού, επιλέγοντας την επιλογή Dataset Description μπορούμε να δούμε αναλυτικά πληροφορίες για τα δεδομένα, όπως προβολή μέρους του Dataset, τους τύπους των στηλών του Dataset και συνολικό αριθμό label κάθε κατηγορίας.



```
-> Dataset:

   LTE/5g Category  Time  Packet Loss Rate  Packet delay  IoT  LTE/5G  GBR  \
0               14     0          0.000001            10    1       0    0
1               18    20          0.001000           100    0       1    1
2               17    14          0.000001           300    0       1    0
3                3    17          0.010000           100    0       1    0
4                9     4          0.010000            50    1       0    0
5               19     2          0.000001            10    1       0    0
6               15     2          0.010000           300    1       0    1
7               19     3          0.001000            50    0       1    0
8                8    20          0.001000           150    0       1    0
9               13    10          0.001000           150    0       1    0

   Non-GBR  AR/VR/Gaming  Healthcare  Industry 4.0  IoT Devices  \
0        1             0           0             0            0
1        0             1           0             0            0
2        1             0           0             0            0
3        1             0           0             0            0
4        1             0           0             0            0
5        1             0           0             1            0
6        0             0           0             0            1
7        1             1           0             0            0
8        1             0           0             0            0
9        1             0           0             0            0

   Public Safety  Smart City & Home  Smart Transportation  Smartphone  \
0              1                  0                     0           0
1              0                  0                     0           0
2              0                  0                     0           1
3              0                  0                     0           1
4              0                  1                     0           0
5              0                  0                     0           0
6              0                  0                     0           0
7              0                  0                     0           0
8              0                  0                     0           1
9              0                  0                     0           1

   slice Type
0           3
1           1
2           1
3           1
4           2
5           3
6           2
7           1
8           1
9           1
```

Περιγραφή του dataset

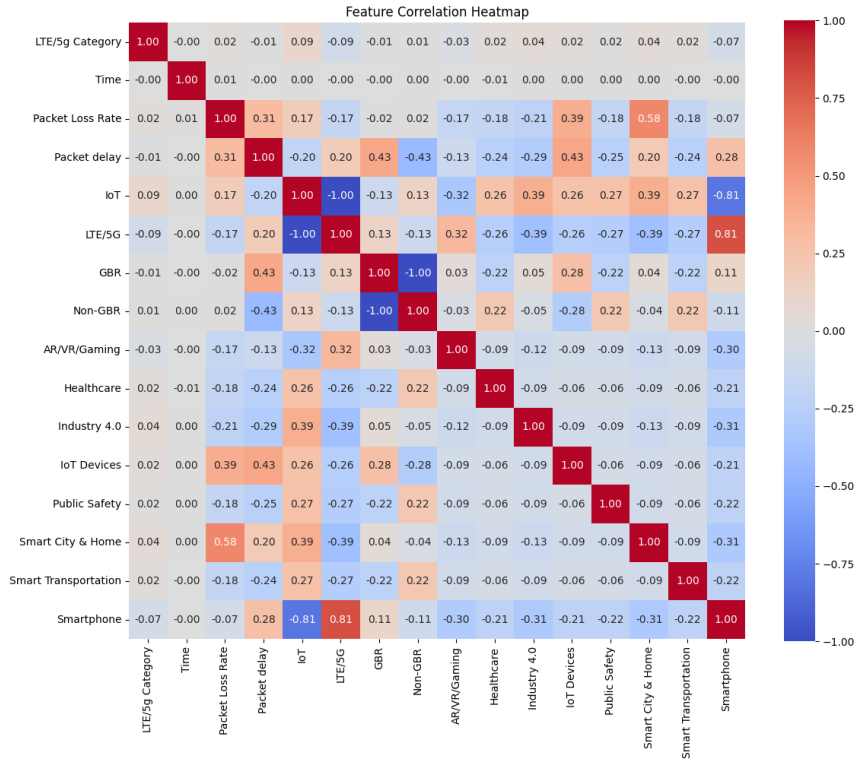
```
-> Dataset Labels (Slice Types):

slice Type
1    16799
3     7392
2     7392
Name: count, dtype: int64
```

Τα labels του dataset

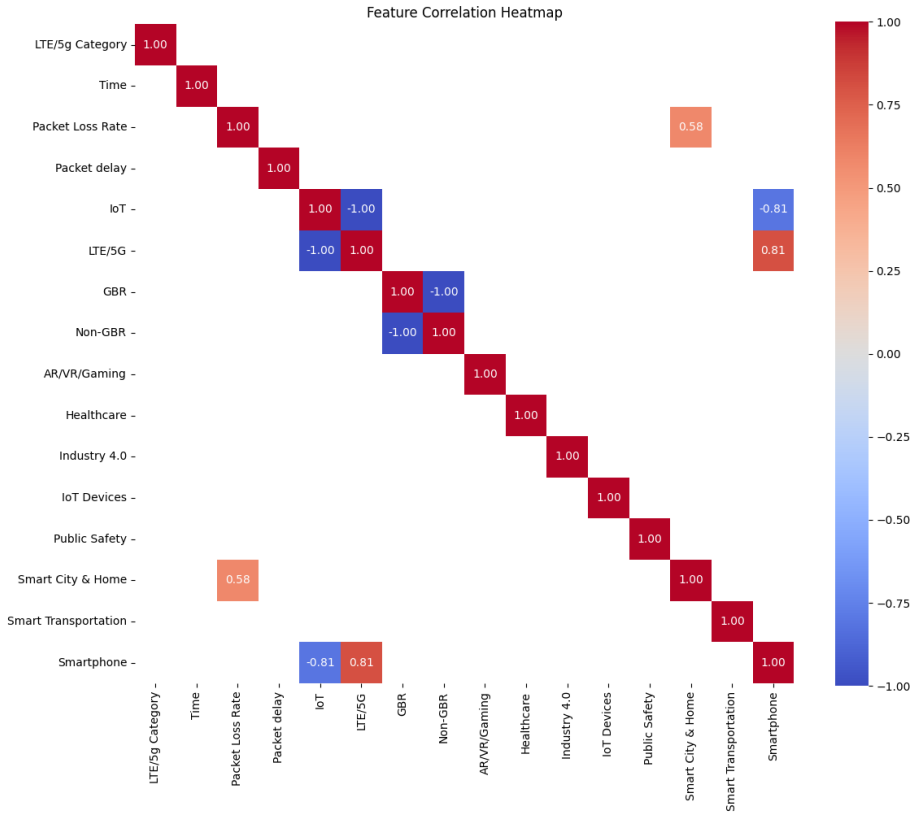
```
-> Dataset Column Types:

LTE/5g Category          int64
Time                     int64
Packet Loss Rate       float64
Packet delay             int64
IoT                      int64
LTE/5G                   int64
GBR                      int64
Non-GBR                  int64
AR/VR/Gaming             int64
Healthcare               int64
Industry 4.0             int64
IoT Devices              int64
Public Safety            int64
Smart City & Home        int64
Smart Transportation     int64
Smartphone               int64
slice Type               int64
dtype: object
```

Τα features του dataset

Επίσης δημιουργούνται δύο Correlation Heatmap που δείχνουν τις συσχετίσεις μεταξύ των label, ένα ολόκληρο και ένα με φιλτραρισμένες τιμές άνω του 0.5. Θετική συσχέτιση δείχνει ότι οι δύο μεταβλητές αυξάνονται μαζί και αρνητική συσχέτιση δείχνει ότι όταν αυξάνεται η μία, μειώνεται η άλλη:



Correlation Heatmap για τα features



Correlation Heatmap για τιμές ανώτερες του 0.5 στα features

Από αυτό το διάγραμμα βλέπουμε ότι υπάρχει ισχυρή θετική συσχέτιση μεταξύ των smartphone και LTE/5G (0.81), και ισχυρή αρνητική συσχέτιση (-0.81) μεταξύ smartphone και IoT. Τέλος βλέπουμε μια μέτρια θετική συσχέτιση μεταξύ των Smart City & Home και Time (0.58).

Κατηγορία κάθε Slice του Dataset

- Slice Type 1: eMBB (Enhanced Mobile Broadband): 1 σε VR/AR και Smartphone.
- Slice Type 2: Massive or Critical Machine Type Communications (mMTC or CMTC) -> 1 σε IoT, Industry 4.0 και Smart City & Home.
- Slice Type 3: Ultra Reliable Low Latency Communications (URLLC) -> 1 σε Public Safety, Healthcare και smart Transportation.

Προεπεξεργασία δεδομένων

Για την πλήρη αξιοποίηση των Deep Neural Networks, στα δεδομένα μας γίνεται Normalization. Με τη κανονικοποίηση των δεδομένων μας, αυξάνουμε την ακρίβεια του μοντέλου μας καθώς και την απόδοσή του. Στη περίπτωσή μας, χρησιμοποιούμε το MinMaxScaling για να περιορίσουμε τα δεδομένα μας ανάμεσα στις τιμές 0 και 1.

```
scaler = MinMaxScaler(feature_range=(0,1))
features = scaler.fit_transform(features)
```

Επίσης για να μπορέσουμε να χειριστούμε τα δεδομένα σωστά, χρησιμοποιούμε τα tensors που χρησιμοποιούν mapping για να εισάγουν τα δεδομένα στο μοντέλο εκμάθησης. Έπειτα, τα εισάγουμε σε ένα TensorDataset και τα φορτώνουμε ως Dataloader datasets στο μοντέλο.

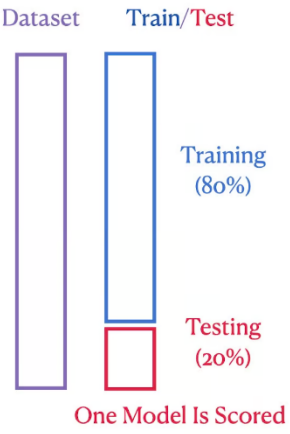
```
train_dataset = TensorDataset(X_train, y_train)
val_dataset = TensorDataset(X_val, y_val)
test_dataset = TensorDataset(X_test, y_test)

batch_size=512

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Διαχωρισμός των Δεδομένων

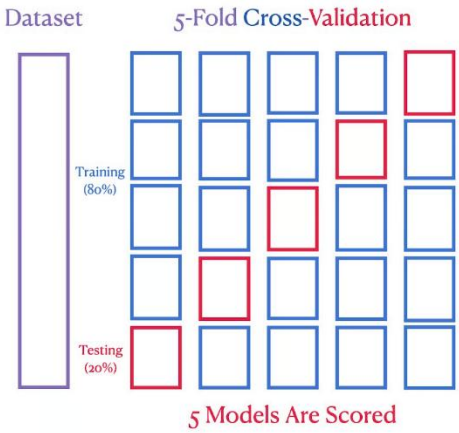
Για να προετοιμάσουμε τα δεδομένα και να αξιολογήσουμε τα μοντέλα που εκπαιδεύσαμε, εφαρμόσαμε την τεχνική του K-Fold Cross Validation. Σε αντίθεση με την μέθοδο Train-Test Split η οποία διαχωρίζει το σύνολο των δεδομένων σε αναλογία 80%-20% (εκπαίδευση, δοκιμή), το K-Fold χωρίζει το dataset σε K ίσα υποσύνολα (folds) και επαναλαμβάνει τη διαδικασία εκπαίδευσης και αξιολόγησης K φορές, κάθε φορά χρησιμοποιώντας διαφορετικό fold ως validation set. Στο πρόγραμμα μας χρησιμοποιήσαμε την τεχνική Stratified K-Fold Cross Validation η οποία είναι μια παραλλαγή που εξασφαλίζει ότι η αναλογία των label παραμένει σταθερή σε κάθε fold. Το Train Test Split χρησιμοποιήθηκε για την εκπαίδευση και την εξαγωγή των τελικών μοντέλων.

```
# Initialize Stratified K-Fold
k_folds = StratifiedKFold(n_splits=k_folds_n, shuffle=True, random_state=42)
```

```
for train_index, val_index in k_folds.split(features_tensor,label_tensor):
    fold_n += 1

    print(f"\n================== Fold {fold_n}/{k_folds_n} ==================")

    # Tensors of every fold
    X_train = features_tensor[train_index]
    y_train = label_tensor[train_index]
    X_val = features_tensor[val_index]
    y_val = label_tensor[val_index]

    # Create fold dataset
    train_fold_dataset = TensorDataset(X_train, y_train)
    val_fold_dataset = TensorDataset(X_val, y_val)
    # Create dataset loader
    train_loader = DataLoader(train_fold_dataset, batch_size=batch_size, shuffle= True)
    val_loader = DataLoader(val_fold_dataset,batch_size=batch_size)
```



Train Test Split



K-Fold Cross Validation

Εξήγηση Classification Report

```
print(classification_report(y_test, test_prediction, digits=2))
```

Precision: Η ακρίβεια του μοντέλου στις θετικές προβλέψεις.

Recall: Πόσο ευαίσθητο είναι ένα μοντέλο (δηλαδή με πόσο μεγάλη ακρίβεια εντοπίζει τα θετικά περιστατικά).

F1-Score: Μια μέση τιμή των Recall και Precision.

Support: Το πλήθος των περιστατικών σε κάθε κλάση.

Deep Neural Networks

Multilayer Perceptron (MLP)

Το MLP είναι ένα απλό Feedforward Neural Network (FNN) στο οποίο η πληροφορία ρέει από την είσοδο προς την έξοδο γραμμικά. Ενδιάμεσα, υπάρχουν κρυφά layers τα οποία εκτελούν πράξεις σε σχέση με την είσοδο και bias νευρώνες οι οποίοι κρίνουν την μετακίνηση του activation function (στη περίπτωσή μας, ReLU) με στόχο την επιτυχημένη εκπαίδευση του μοντέλου.

```
# Load the model and parameters
model = MLP(input_size=16, hidden_units=32, dropout=0.3, num_classes=3).to(device)
optimizer = optim.Adam(model.parameters(), lr=0.001,weight_decay=0.0001)
criterion = nn.CrossEntropyLoss()
```

```python
class MLP(nn.Module):
    def __init__(self, input_size, hidden_units, dropout, num_classes):
        super(MLP, self).__init__()

        # Hidden layer
        self.fc1 = nn.Linear(input_size, hidden_units)

        # Dropout to avoid overfitting
        self.dropout = nn.Dropout(dropout)

        # Output layer
        self.fc2 = nn.Linear(hidden_units, num_classes)

    def forward(self, x):
        # Fully connected layer and ReLU
        x = F.relu(self.fc1(x))

        # Dropout
        x = self.dropout(x)

        # Final Fully connected layer
        x = self.fc2(x)

        return x
```

Multilayer Perceptron

Στο MLP του συγκεκριμένου κώδικα, έχουν εισαχθεί και τα dropout layers, τα οποία αφαιρούν νευρώνες σταδιακά με κάθε εκπαίδευση, με σκοπό την αποφυγή του overfitting. Το μοντέλο περιέχει ένα hidden layer.Στο Forward Propagation χρησιμοποιούμε ReLU(Rectified Linear Unit).

Convolutional Neural Network (CNN)

Τα CNN μοντέλα, που χρησιμοποιούνται κυρίως για εικόνες και βίντεο, ή πιο γενικά σε δισδιάστατα δεδομένα, ενώ είναι ένα είδος FNN, λειτουργούν διαφορετικά από τα MLP μοντέλα. Πιο συγκεκριμένα, εκτελούν τέσσερα συγκεκριμένα layers που τα κάνουν να διαφέρουν από τα υπόλοιπα μοντέλα:

Convolutional layers

Με την χρήση αυτών των layers, εφαρμόζει φίλτρα (kernels) πάνω στα δεδομένα, και το αποτέλεσμα που παίρνουμε είναι το Feature map, το οποίο χρησιμοποιούμε για να εντοπίσουμε τα features των δεδομένων και ταυτόχρονα να αναγνωρίσουμε σχήματα και μοτίβα σε εικόνες ή βίντεο.

Pooling layers

Η διαδικασία που ονομάζεται pooling είναι η απλοποίηση και σμίκρυνση των δεδομένων του Feature map με στόχο το dimensionality reduction, δηλαδή τη μείωση των input features στο μοντέλο μας.

Flattening layers

Αυτά τα layers χρησιμοποιούνται για την αλλαγή διάστασης των δεδομένων από 2D σε 1D.

Dense layers

Τα layers αυτά χρησιμοποιούνται για την πρόβλεψη του μοντέλου.

```
# Load the model and parameters
model = CNN1D(input_size=16, hidden_units=32, dropout=0.3, num_classes=3).to(device)
optimizer = optim.Adam(model.parameters(), lr=0.001,weight_decay=0.0001)
criterion = nn.CrossEntropyLoss()
```

```
class CNN1D(nn.Module):
    def __init__(self, input_size, hidden_units, dropout, num_classes):
        super(CNN1D, self).__init__()

        # Convolutional 1D
        self.conv1 = nn.Conv1d(
            in_channels=1, out_channels=16, kernel_size=3, stride=1, padding=1)

        # Max Pooling layer
        self.pool = nn.MaxPool1d(kernel_size=2, stride=2)

        # Hidden layer
        self.fc1 = nn.Linear(input_size * 8, hidden_units)

        # Dropout to avoid overfitting
        self.dropout = nn.Dropout(dropout)

        # Output layer
        self.fc2 = nn.Linear(hidden_units, num_classes)

    def forward(self, x):

        # Convolution -> ReLU -> MaxPooling
        x = self.pool(F.relu(self.conv1(x)))

        # Flatten to 2D
        x = x.view(x.size(0), -1)

        # Fully connected layer and ReLU
        x = F.relu(self.fc1(x))

        # Dropout
        x = self.dropout(x)

        # Final Fully connected layer
        x = self.fc2(x)

        return x
```

Convolutional Neural Network

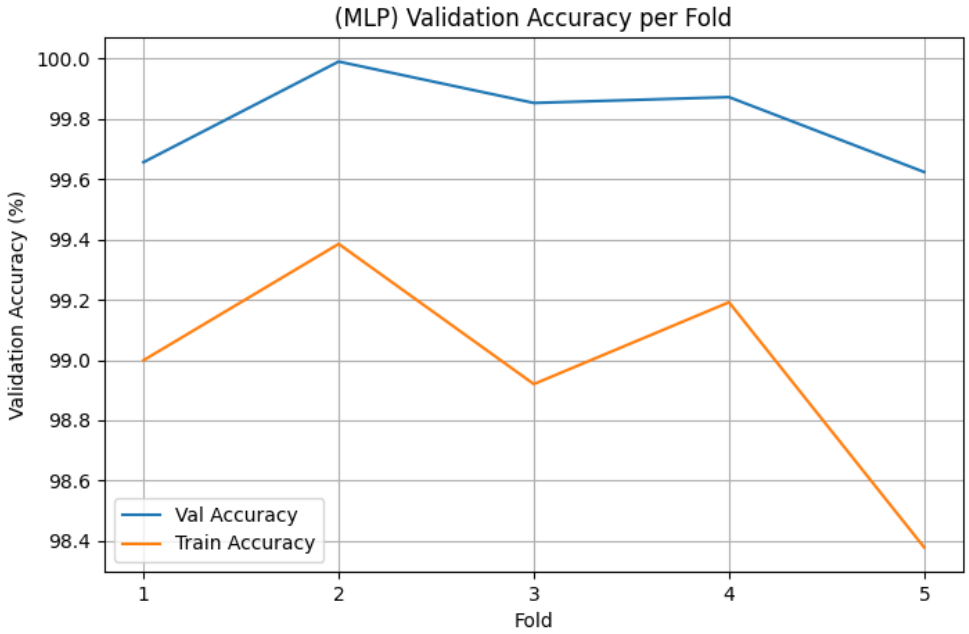
Στο συγκεκριμένο μοντέλο, χρησιμοποιείται το Conv1D καθώς εισάγονται δεδομένα τύπου float και int, οπότε δε χρειάζεται να χρησιμοποιηθεί το 2D. Στο στιγμιότυπο φαίνονται και τα layers του μοντέλου(pooling, hidden, dropout, output, flattening, dense).

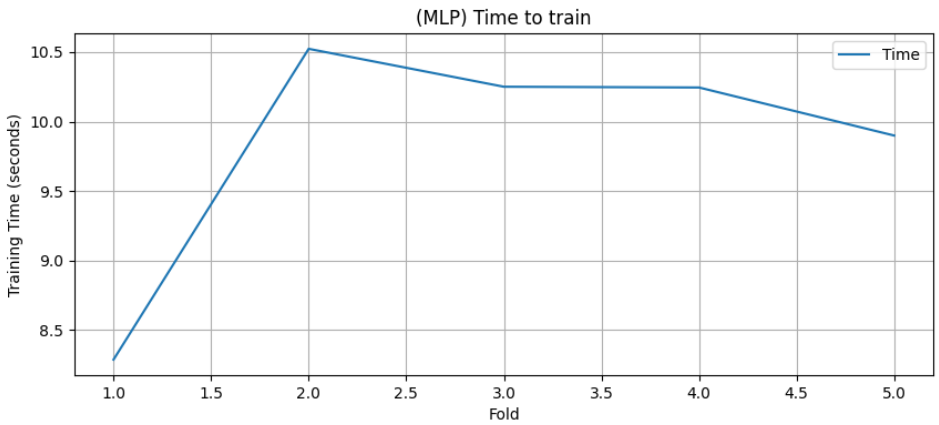
Αποτελέσματα Deep Neural Networks

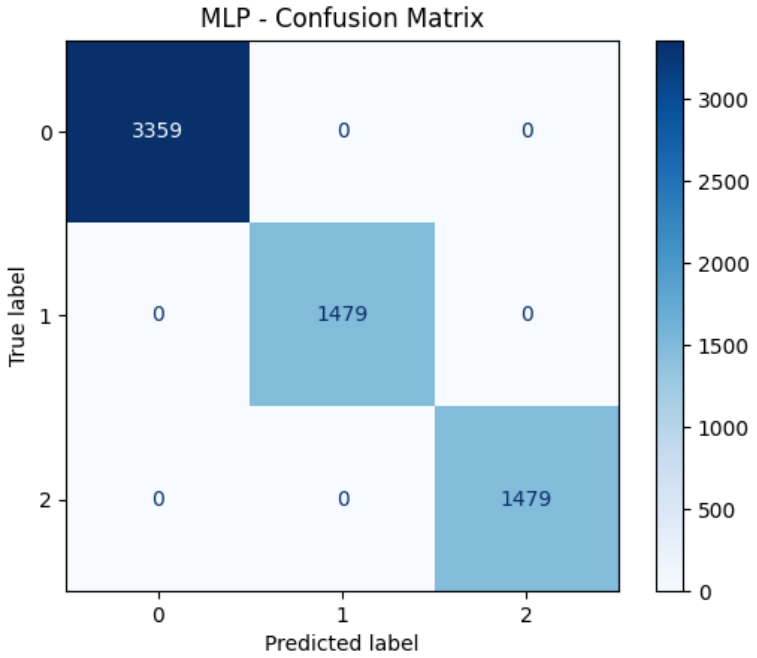
MLP

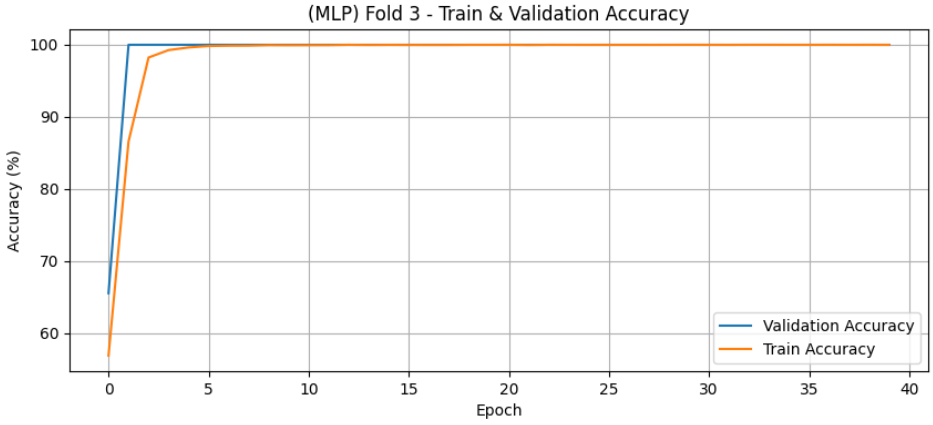
Αποτελέσματα K-Fold Cross Validation

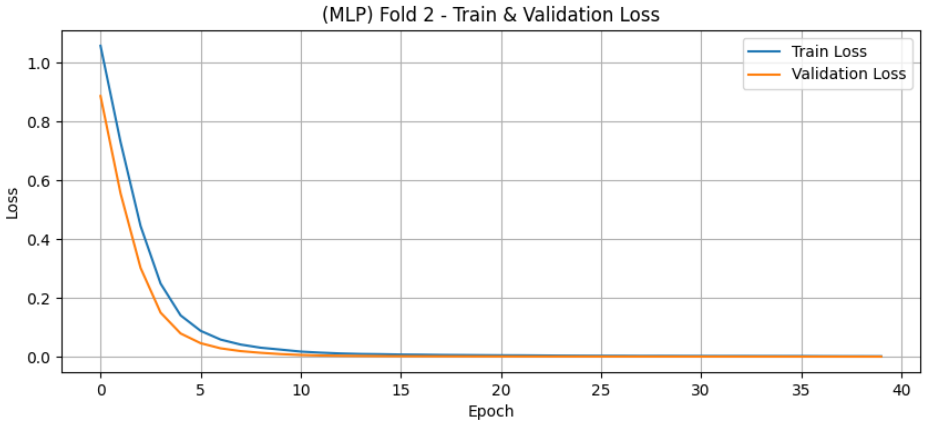
```
-> Average K-Fold Train Accuracy: 98.97%

-> Average K-Fold Validation Accuracy: 99.80%

-> Average MLP Training Time: 9.84 seconds
```





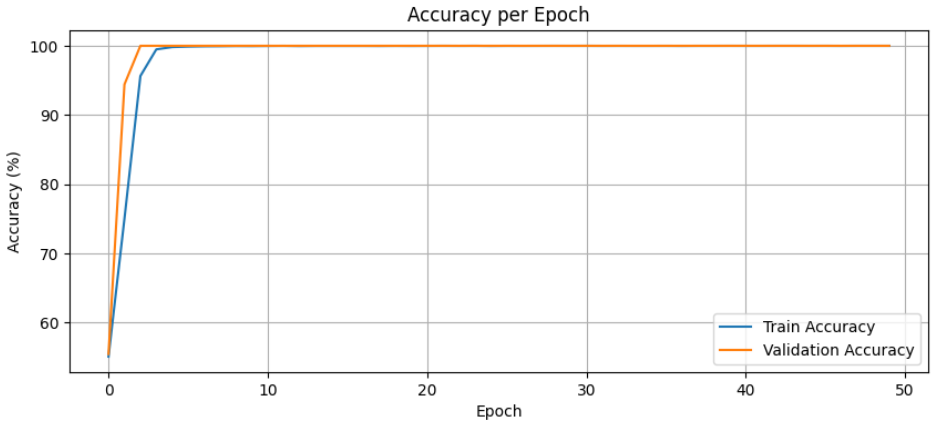

MLP - Confusion Matrix
3359
0
0
0
1479
0
0
0
1479
True label
Predicted label
0
1
2
3000
2500
2000
1500
1000
500
0

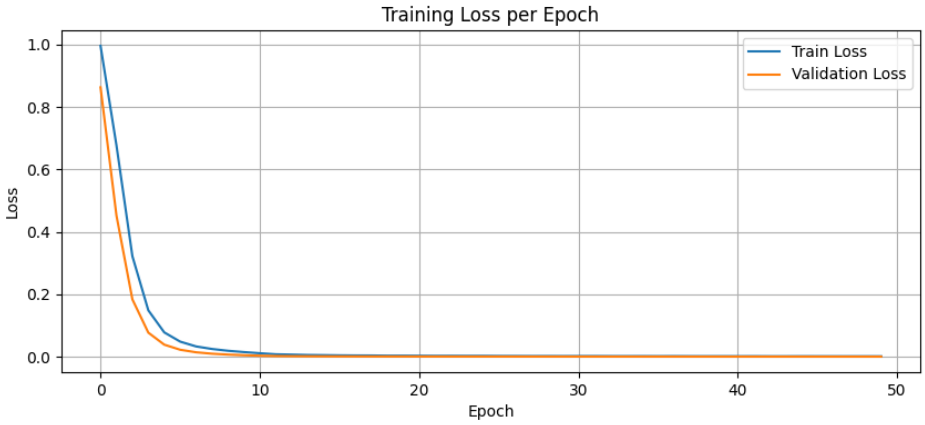

(MLP) Fold 3 - Train & Validation Accuracy
Accuracy (%)
Epoch
Validation Accuracy
Train Accuracy

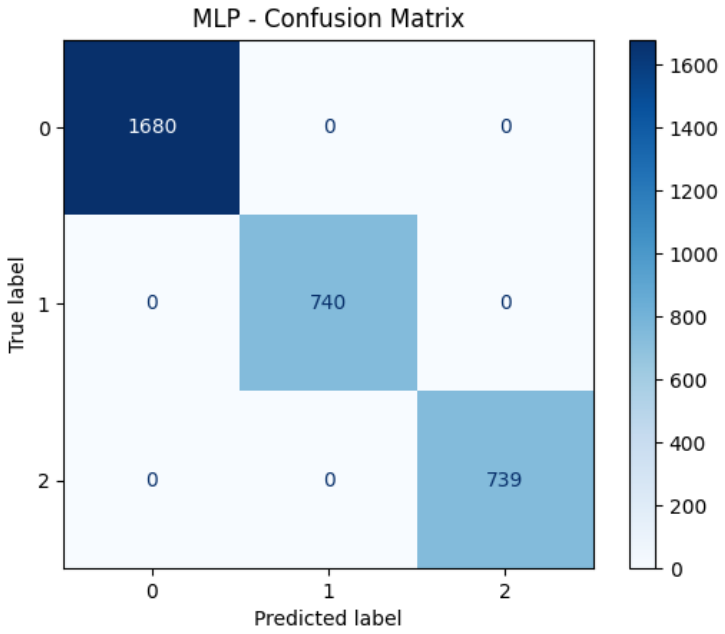

(MLP) Fold 2 - Train & Validation Loss
Loss
Epoch
Train Loss
Validation Loss

Train Test Spit

```
==============================

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1680
           1       1.00      1.00      1.00       740
           2       1.00      1.00      1.00       739

    accuracy                           1.00      3159
   macro avg       1.00      1.00      1.00      3159
weighted avg       1.00      1.00      1.00      3159

==============================

-> Test Accuracy: 100.00%

==============================
```



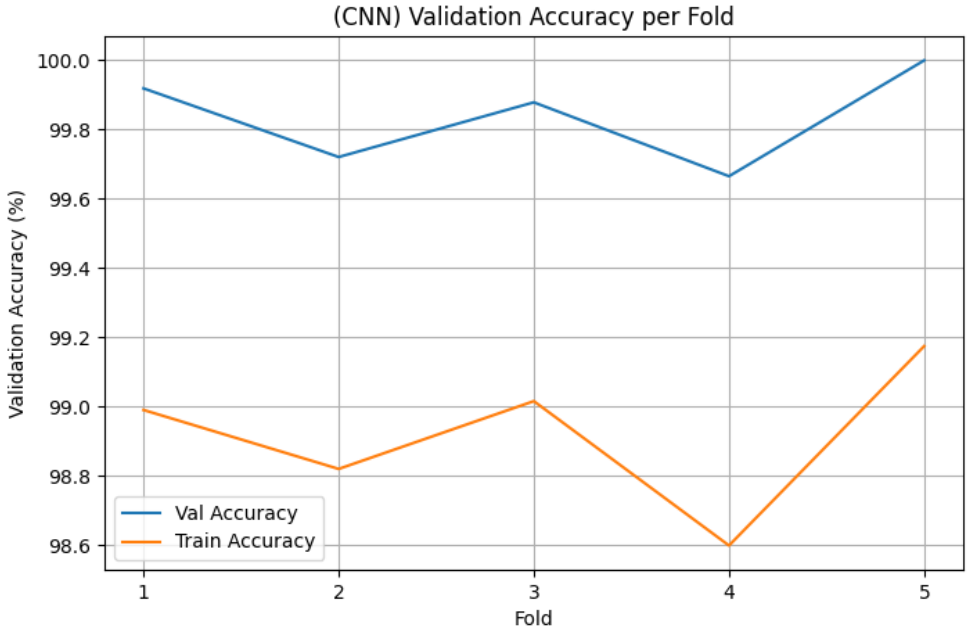


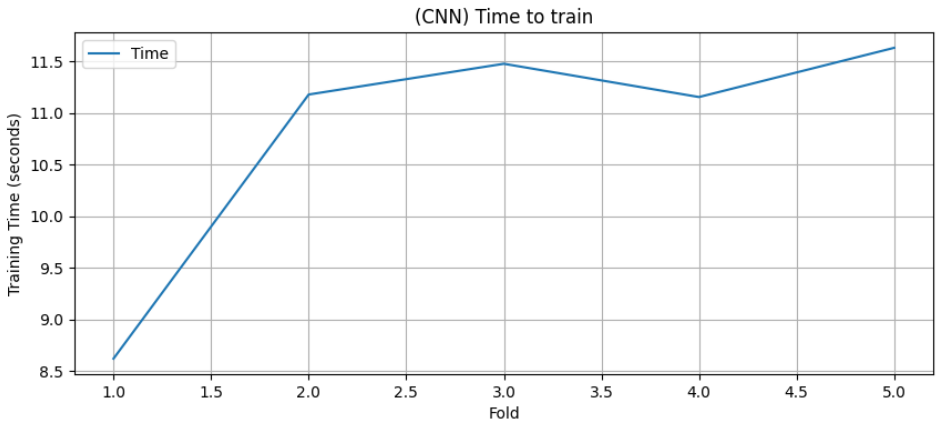

MLP - Confusion Matrix
True label
Predicted label
0
1
2
1680
0
0
0
740
0
0
0
739
1600
1400
1200
1000
800
600
400
200
0

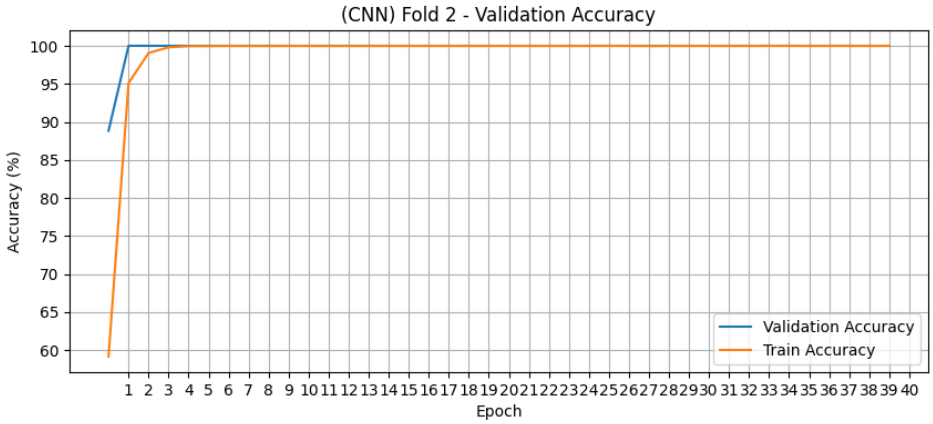
CNN

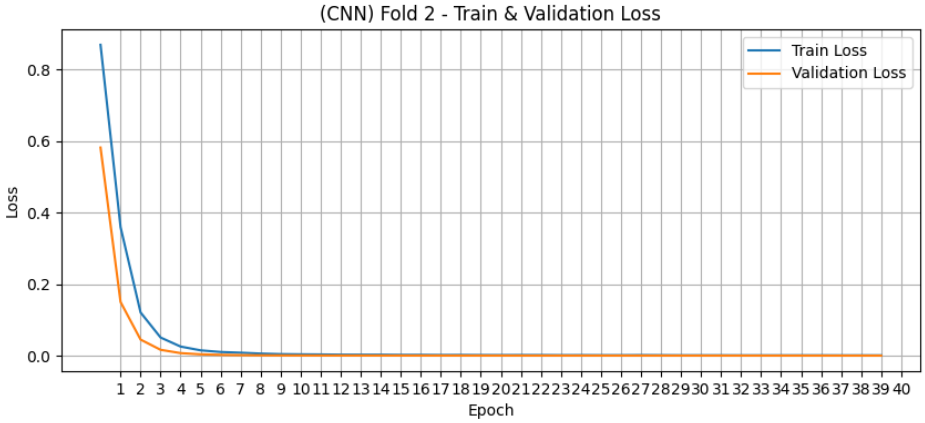
Αποτελέσματα K-Fold Cross Validation

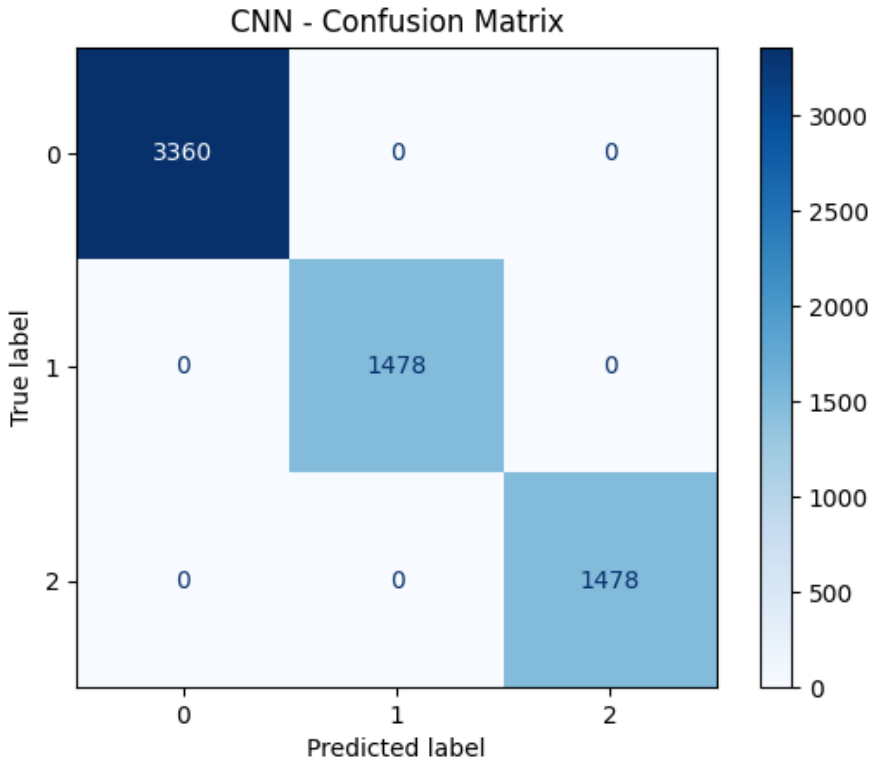
```
-> Average K-Fold Train Accuracy: 98.92%

-> Average K-Fold Validation Accuracy: 99.84%

-> Average CNN Training Time: 10.81 seconds
```





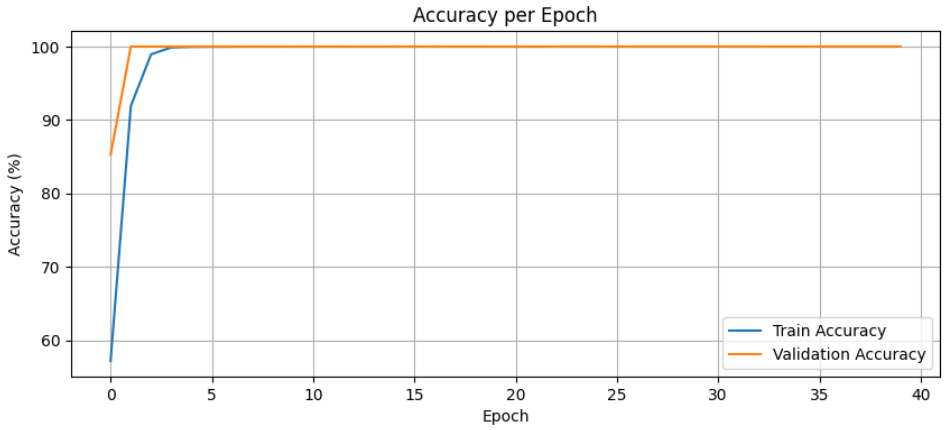

(CNN) Fold 2 - Validation Accuracy
Accuracy (%)
Epoch
Validation Accuracy
Train Accuracy

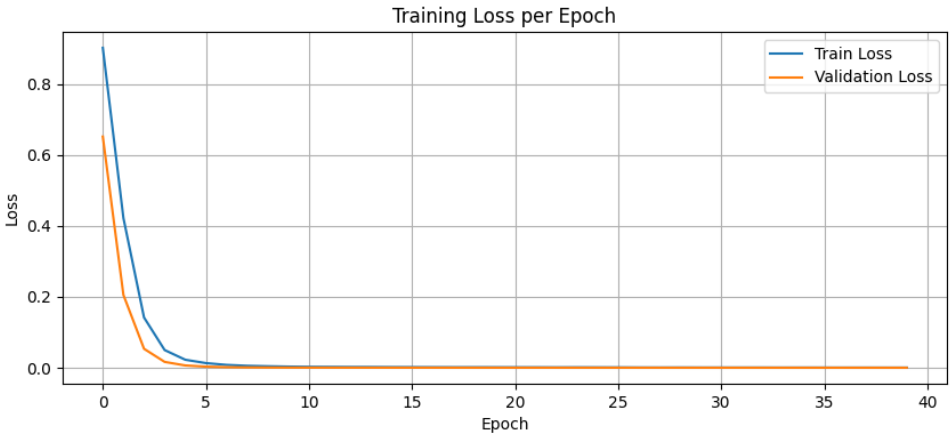

(CNN) Fold 2 - Train & Validation Loss
Loss
Epoch
Train Loss
Validation Loss

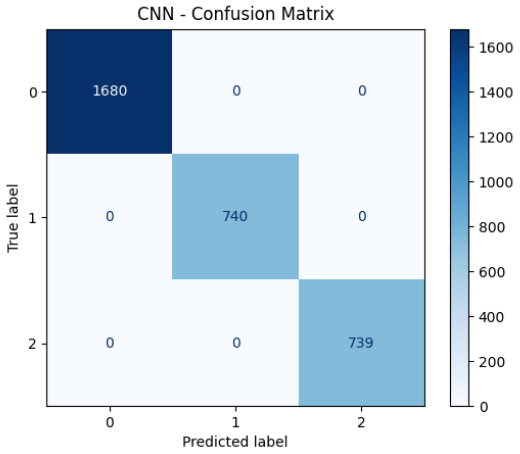

CNN - Confusion Matrix
True label
Predicted label
3360
0
0
0
1478
0
0
0
1478

Train Test Spit

```
================================

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1680
           1       1.00      1.00      1.00       740
           2       1.00      1.00      1.00       739

    accuracy                           1.00      3159
   macro avg       1.00      1.00      1.00      3159
weighted avg       1.00      1.00      1.00      3159

================================

-> Test Accuracy: 100.00%

================================
```







Σχολιασμός αποτελεσμάτων

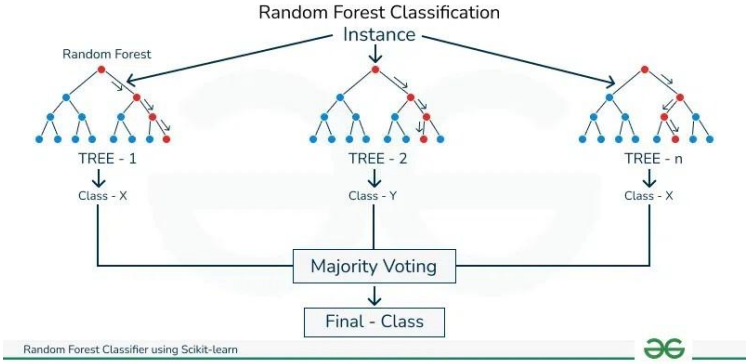
Με τα παραπάνω αποτελέσματα βγάζουμε το συμπέρασμα πως και τα δύο μας μοντέλα είναι εξίσου ισχυρά και προβλέπουν με ακρίβεια το slice Type στο οποίο ανήκει κάθε sample. Η υψηλή ακρίβεια των μοντέλων μας οφείλεται στο μικρό πλήθος των δεδομένων μας , το μικρό πλήθος labels (1,2 και 3) και στην απλότητα των δεδομένων (οι περισσότερες κατηγορίες έχουν μόνο τις τιμές 0 και 1). Τέλος ο χρόνος εκπαίδευσης κάθε fold είναι παρόμοιος και στα δύο μοντέλα. Η πιθανότητα overfitting εξαλείφεται με τη χρήση των dropout layers, του μικρού learning rate καθώς και των αποτελεσμάτων του K-Fold Cross Validation και στα δύο μοντέλα.

Απλά μοντέλα μηχανικής εκμάθησης

Υλοποιήθηκαν και τα παρακάτω απλά μοντέλα για να ελεγχθεί η ακρίβεια τους καθώς και η ταχύτητα πρόβλεψής τους σε σύγκριση με τα Deep Neural Networks:

Random Forest Classifier

Πρόκειται για πολλαπλά δέντρα απόφασης σε διαφορετικά υποσύνολα των δεδομένων. Κάθε δεδομένο εκπαιδεύεται ανεξάρτητα σε τυχαίο υποσύνολο και η πλειοψηφία των ψήφων καθορίζει το αποτέλεσμα της πρόβλεψης του μοντέλου.

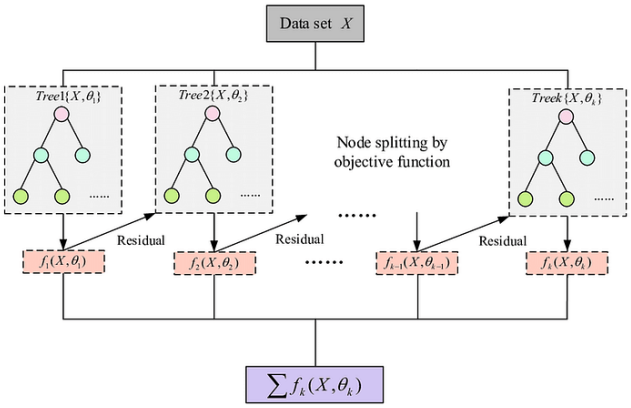


Random Forest Classifier

```python
# Initialize model
randomForestModel = RandomForestClassifier(n_estimators=100, random_state=42)

# Training model
randomForestModel.fit(X_train,y_train)

# Predictions of model
train_predictions = randomForestModel.predict(X_train)
test_predictions = randomForestModel.predict(X_test)
val_predictions = randomForestModel.predict(X_val)
```

XGBoost Classifier

Εκπαιδεύεται σε decision trees τα οποία εκτελούνται σειριακά. Κάθε δέντρο απόφασης εκπαιδεύεται από τα αποτελέσματα των προηγούμενων και το αποτέλεσμα προκύπτει από το άθροισμα όλων των αποτελεσμάτων των δέντρων απόφασης.



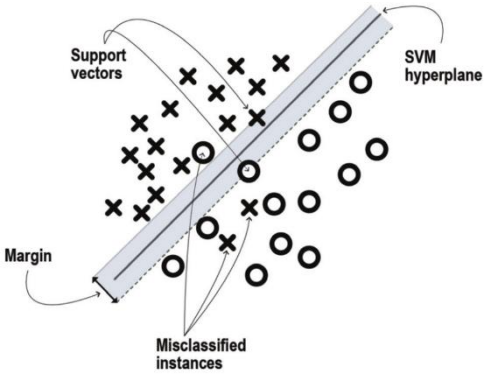
XGBoost Classifier

```python
# Initialize Model
model = xgb.train(
    parameters,
    train_dmatrix,
    num_boost_round=100,
    evals=[(train_dmatrix,'train'),(val_dmatrix,'validation')],
    early_stopping_rounds=10,
    evals_result= evals_res,
    verbose_eval = True
)
```

```python
# Predictions of model
train_predictions = np.argmax(model.predict(train_dmatrix), axis=1)
test_predictions = np.argmax(model.predict(test_dmatrix), axis=1)
val_predictions = np.argmax(model.predict(val_dmatrix), axis=1)
```

LinearSVC

Το LinearSVC βρίσκει το υπερεπίπεδο που διαχωρίζει τις κατηγορίες με το μέγιστο περιθώριο και χρησιμοποιεί γραμμικό πυρήνα για να προβλέψει τις κλάσεις των labels.



LinearSVC

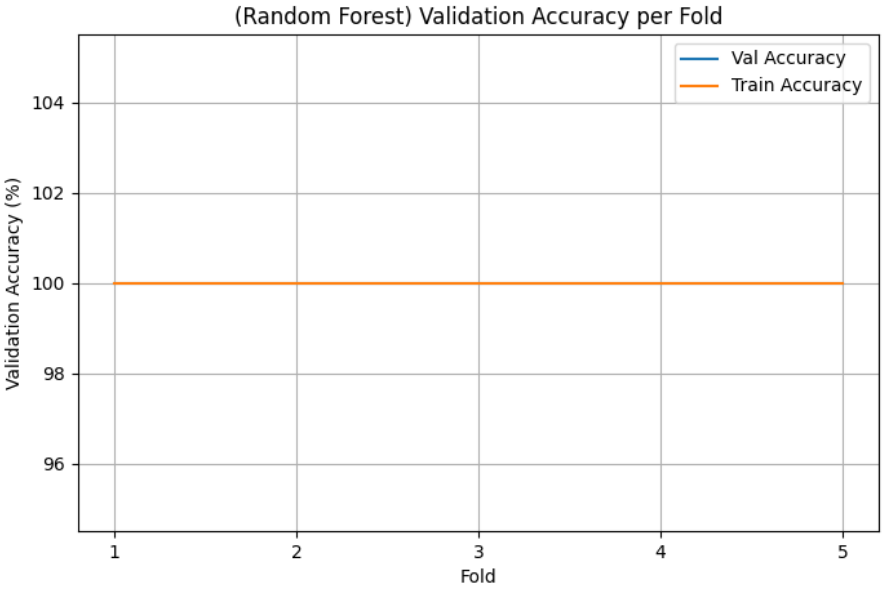
```
# Initialize model
linearSVC = SVC(kernel='linear', probability=True, random_state=42)

# Training model
linearSVC.fit(X_train,y_train)

# Predictions of model
train_predictions = linearSVC.predict(X_train)
test_predictions = linearSVC.predict(X_test)
val_predictions = linearSVC.predict(X_val)
```

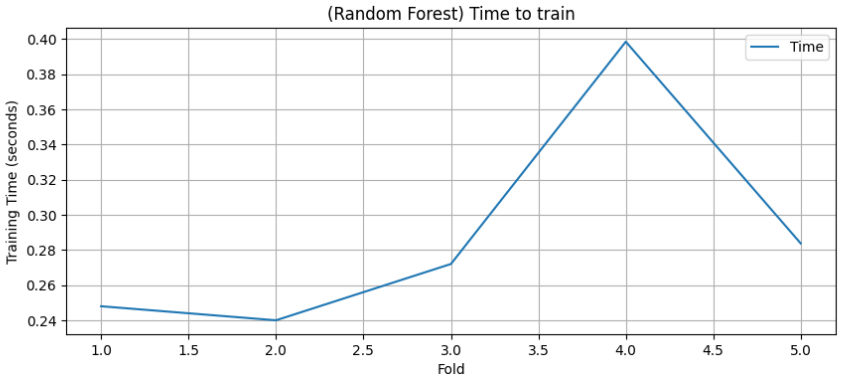
Αποτελέσματα Machine Learning

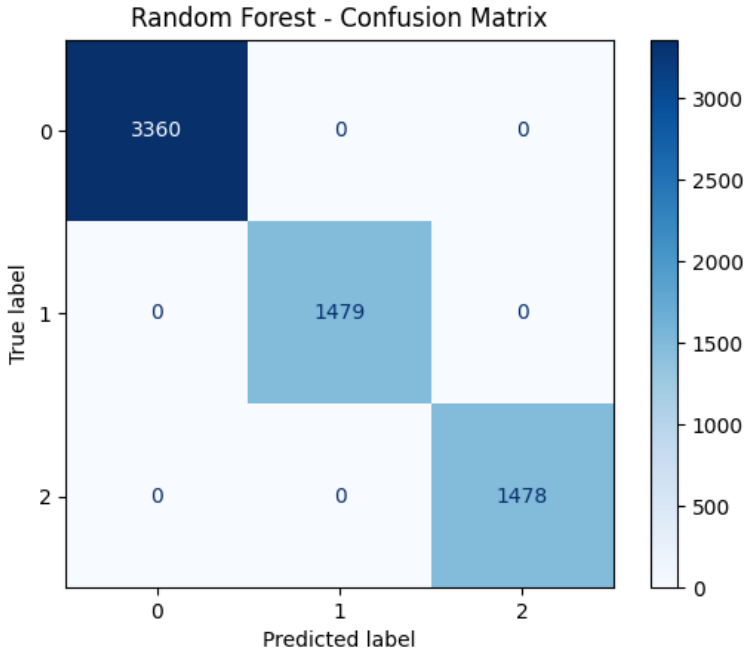
Random Forest Classifier

Αποτελέσματα K-Fold Cross Validation

```
-> Average Validation Accuracy: 100.00%

-> Average Train Accuracy: 100.00%
```

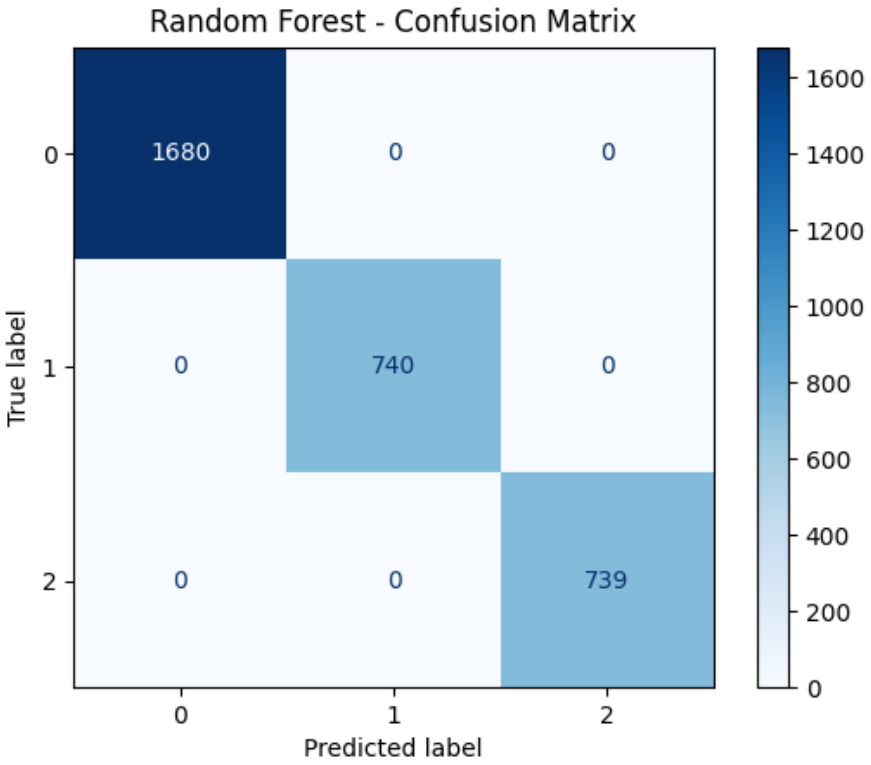

(Random Forest) Validation Accuracy per Fold
Val Accuracy
Train Accuracy
Validation Accuracy (%)
104
102
100
98
96
1
2
3
4
5
Fold


(Random Forest) Time to train
Time
Training Time (seconds)
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fold


Random Forest - Confusion Matrix
3360
0
0
0
1479
0
0
0
1478
True label
Predicted label
0
1
2
3000
2500
2000
1500
1000
500
0

Train Test Spit

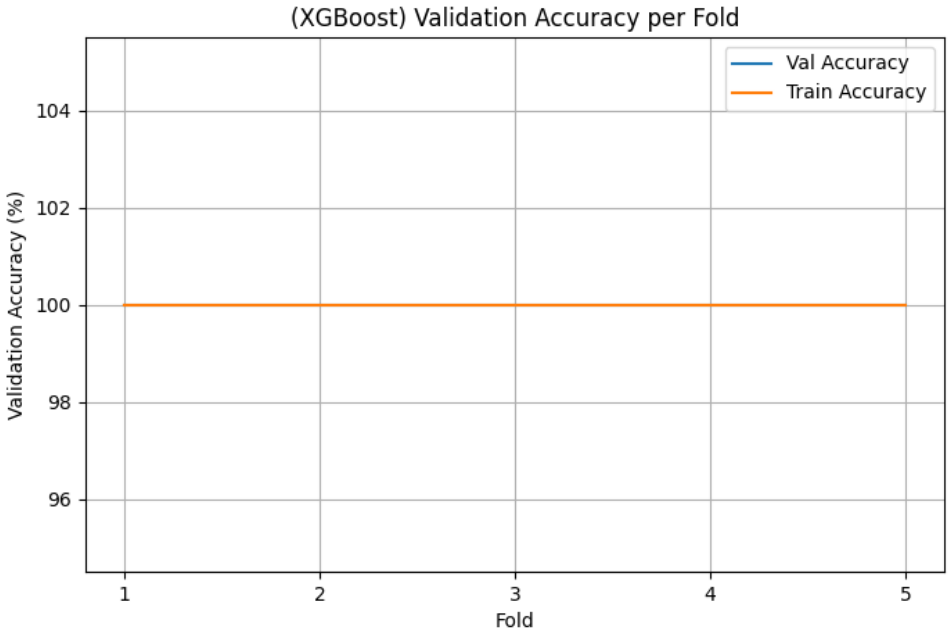
```
================================

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1680
           1       1.00      1.00      1.00       740
           2       1.00      1.00      1.00       739

    accuracy                           1.00      3159
   macro avg       1.00      1.00      1.00      3159
weighted avg       1.00      1.00      1.00      3159

================================

-> Train Accuracy: 100.00%

-> Val Accuracy: 100.00%

-> Test Accuracy: 100.00%

================================
```

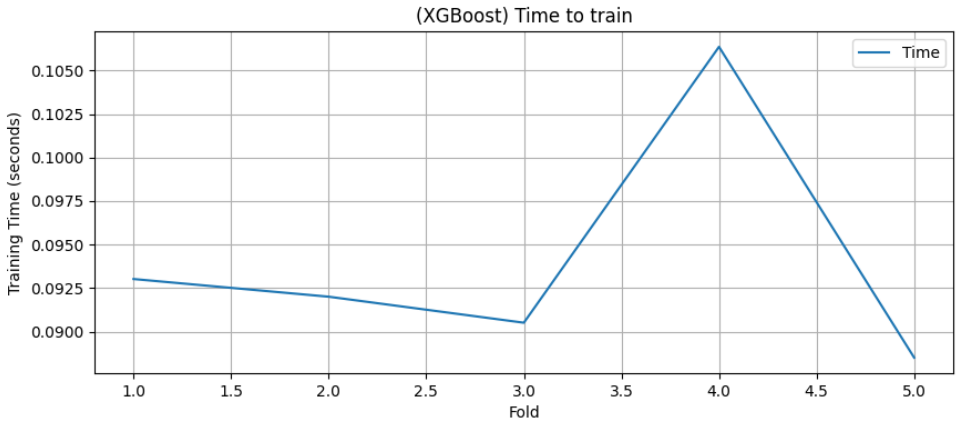


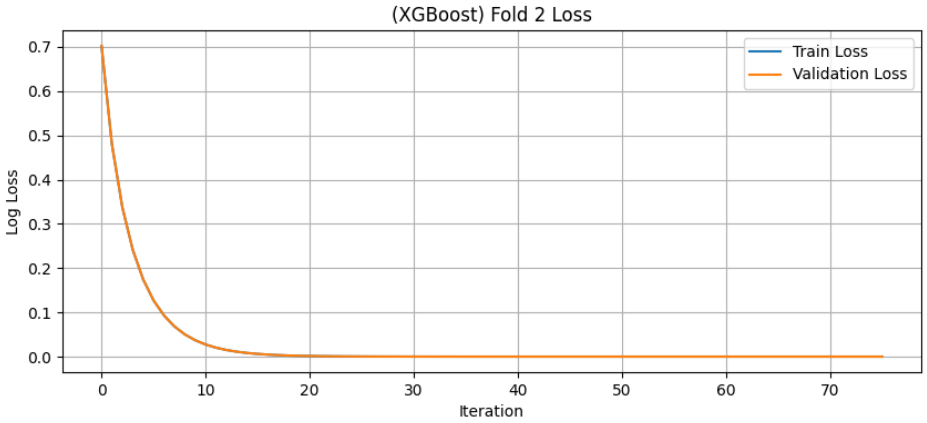
XGBoost Classifier

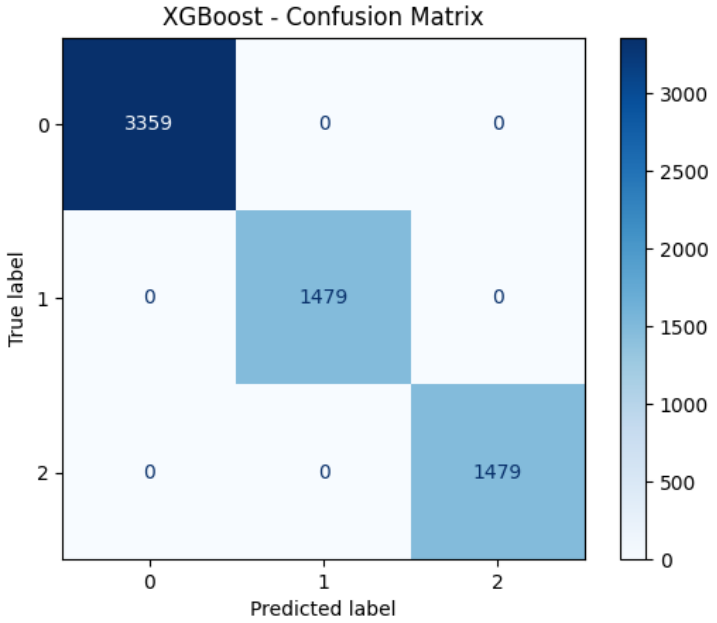
Αποτελέσματα K-Fold Cross Validation

```
-> Average Validation Accuracy: 100.00%

-> Average Train Accuracy: 100.00%

-> Average Validation Loss: 0.0324837131

-> Average Train Loss: 0.0324836923
```





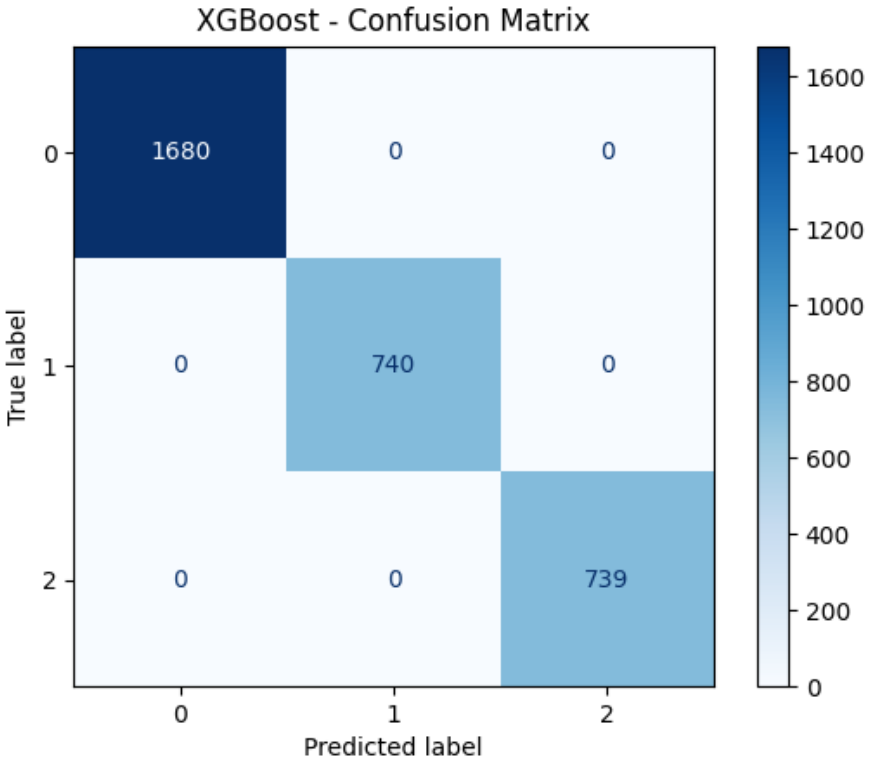




Train Test Spit

```
[0]     train-mlogloss:0.70238  validation-mlogloss:0.70238
[1]     train-mlogloss:0.47982  validation-mlogloss:0.47982
[2]     train-mlogloss:0.33758  validation-mlogloss:0.33758
[3]     train-mlogloss:0.24152  validation-mlogloss:0.24152
[4]     train-mlogloss:0.17463  validation-mlogloss:0.17463
[5]     train-mlogloss:0.12717  validation-mlogloss:0.12717
[6]     train-mlogloss:0.09306  validation-mlogloss:0.09306
[7]     train-mlogloss:0.06833  validation-mlogloss:0.06833
[8]     train-mlogloss:0.05031  validation-mlogloss:0.05031
```

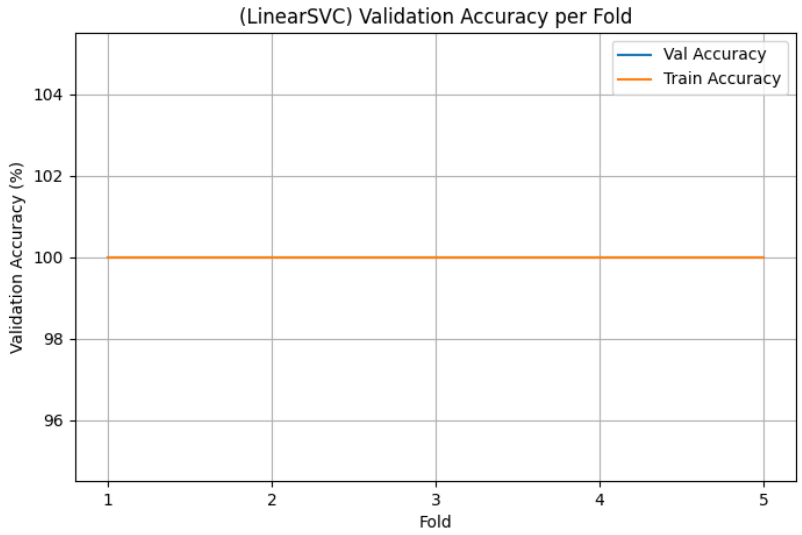
```
================================

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1680
           1       1.00      1.00      1.00       740
           2       1.00      1.00      1.00       739

    accuracy                           1.00      3159
   macro avg       1.00      1.00      1.00      3159
weighted avg       1.00      1.00      1.00      3159

================================

-> Train Accuracy: 100.00%

-> Val Accuracy: 100.00%

-> Test Accuracy: 100.00%

================================
```

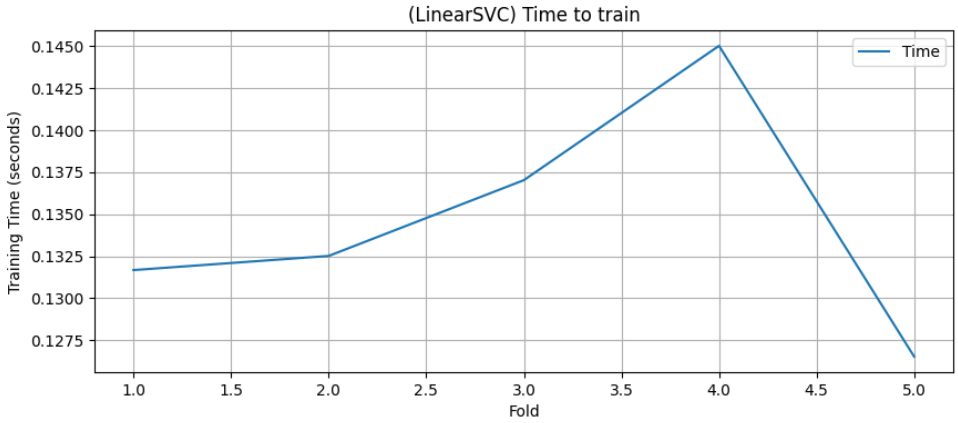


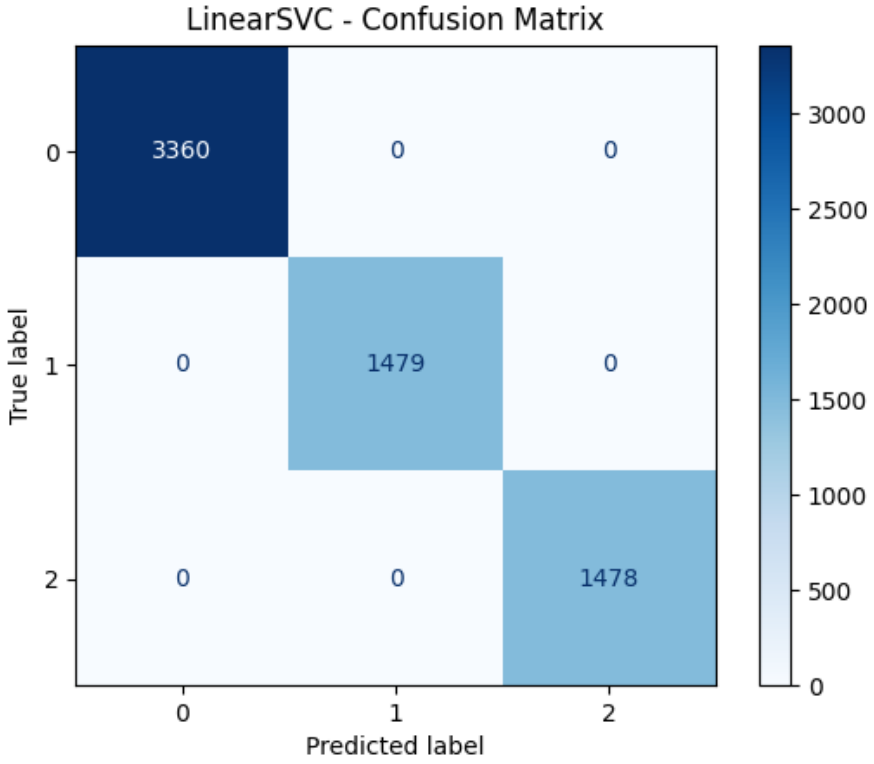
Linear SVC

Αποτελέσματα K-Fold Cross Validation

```
-> Average Validation Accuracy: 100.00%

-> Average Train Accuracy: 100.00%
```

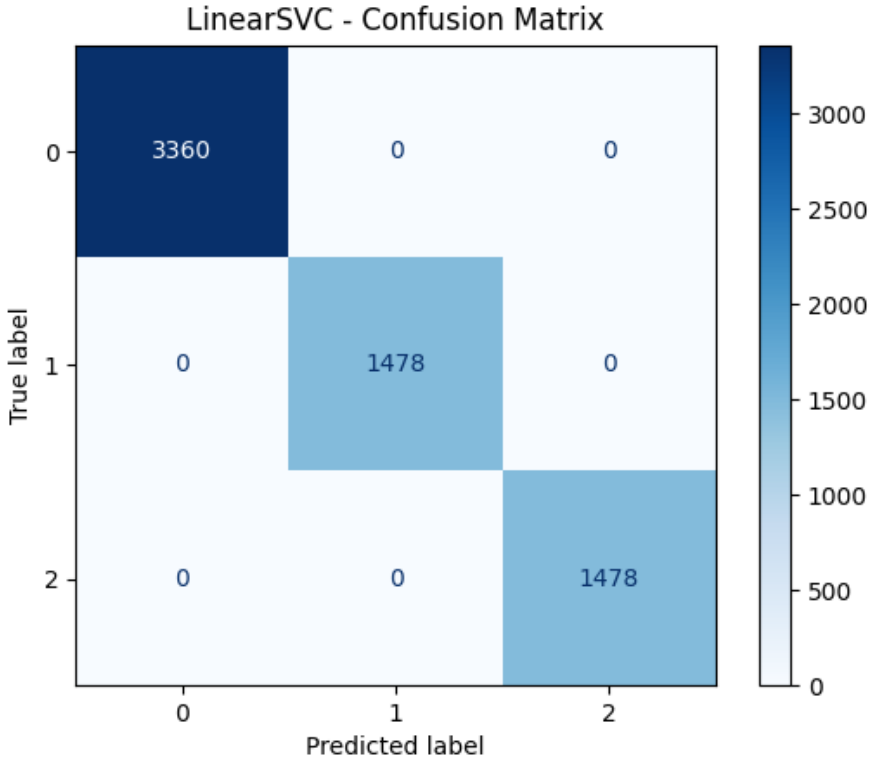






Train Test Spit

```
================================

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1680
           1       1.00      1.00      1.00       740
           2       1.00      1.00      1.00       739

    accuracy                           1.00      3159
   macro avg       1.00      1.00      1.00      3159
weighted avg       1.00      1.00      1.00      3159

================================

-> Train Accuracy: 100.00%

-> Val Accuracy: 100.00%

-> Test Accuracy: 100.00%

================================
```



Σχολιασμός Αποτελεσμάτων

Και στα τρία μοντέλα παρατηρήθηκε υψηλή ταχύτητα εκπαίδευσης και ακρίβειας. Η διαφορά στην απόδοση σε σχέση με τα Deep Neural Networks είναι αμελητέα, γεγονός που οφείλεται, όπως αναφέρθηκε προηγουμένως, στην απλότητα των δεδομένων και στα εύκολα διαχωρίσιμες κατηγορίες. Η μεγάλη διαφορά σε σχέση με τα Deep Neural Networks είναι οι χρόνοι που έως εκατό φορές πιο γρήγορα στην εκπαίδευση.

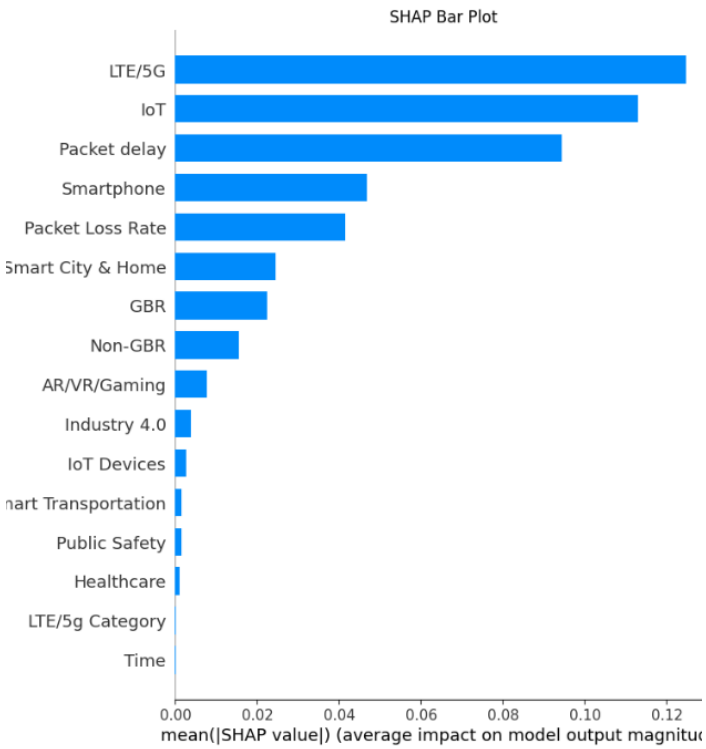
Explainability AI(XAI)

Με τη χρήση του XAI, μπορούμε να αναγνωρίσουμε ποια features του dataset είναι τα πιο σημαντικά για την σωστή πρόβλεψη του slice Type. Για την υλοποίηση του XAI, χρησιμοποιήθηκε το SHAP για την αναγνώριση της βαρύτητας κάθε feature στο Random Forest Classifier.

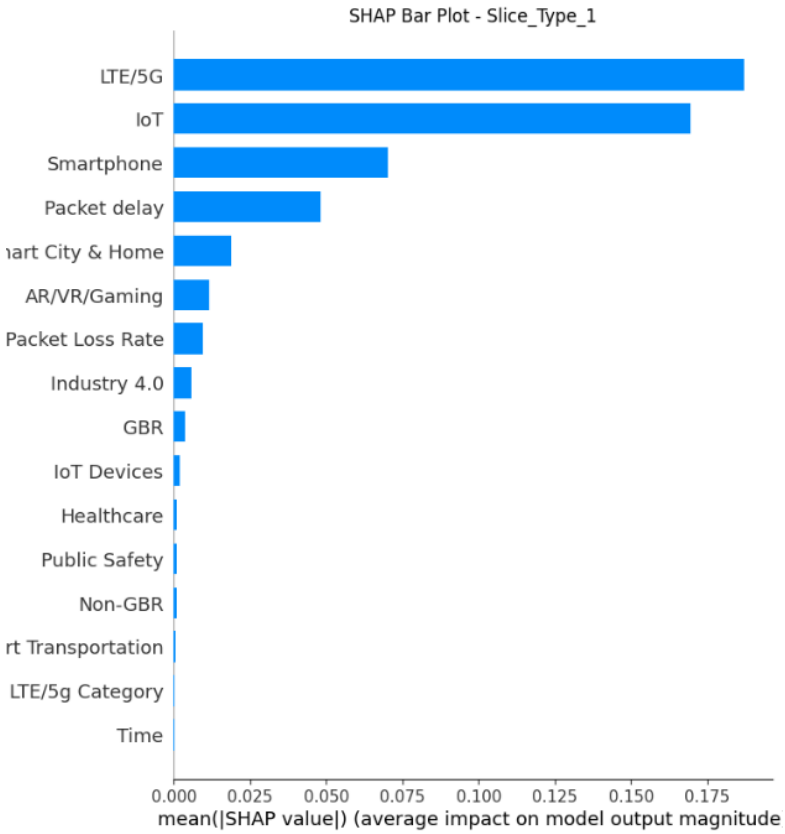
SHAP

Το SHAP (Shapley Additive exPlanations) είναι μια προσέγγιση που βασίζεται σε θεωρία παιχνιδιού με σκοπό την παρουσίαση της εξόδου του μοντέλου μηχανικής εκμάθησης καθώς και την σημασία κάθε feature στο τελικό αποτέλεσμα.

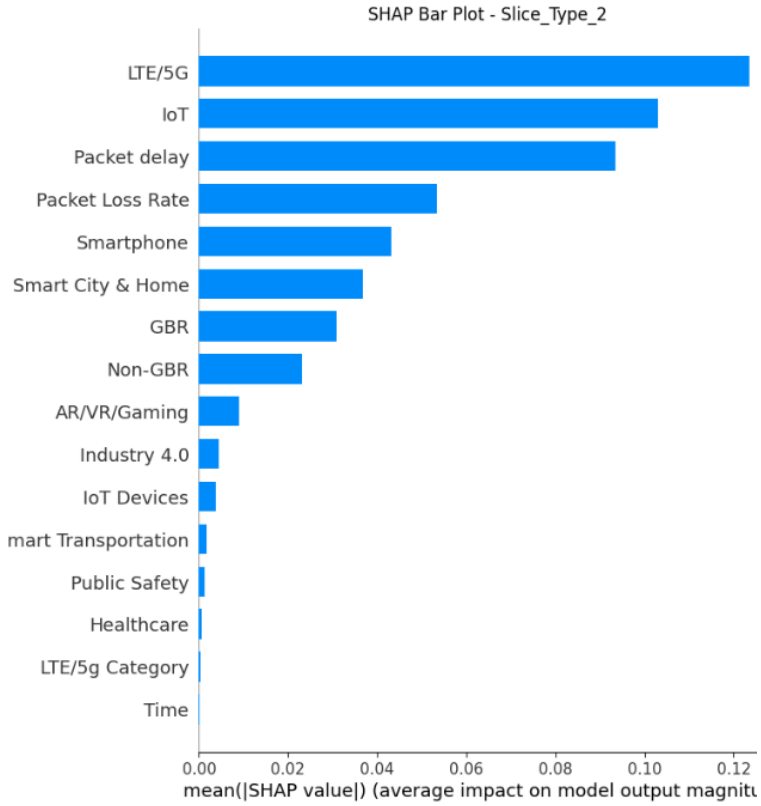
```python
# SHAP Calculation
print("-> Running SHAP...\n")

# Build Tree explainer
explainer = shap.TreeExplainer(model)

# Estimate SHAP values
shap_values = explainer.shap_values(features_scaled_df)
```



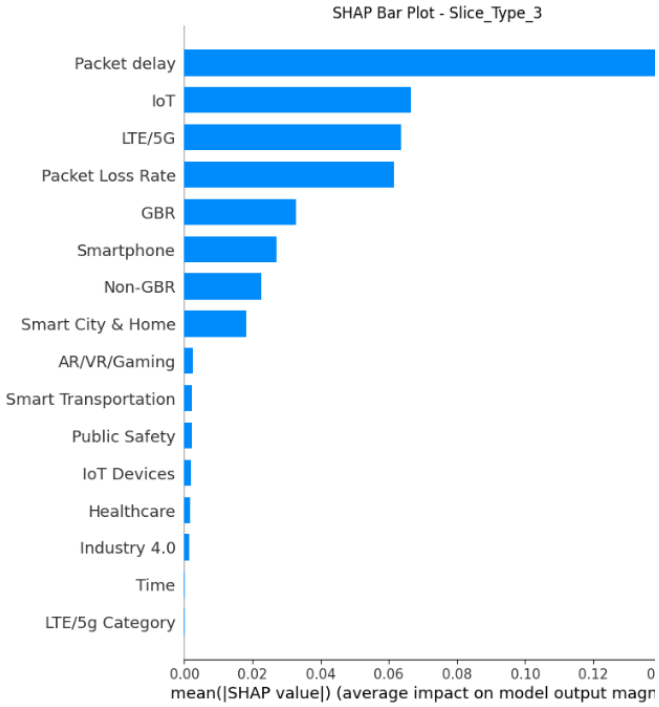
Γενικό SHAP Bar Plot



SHAP Bar Plot για το Slice Type 1



SHAP Bar Plot για το Slice Type 2

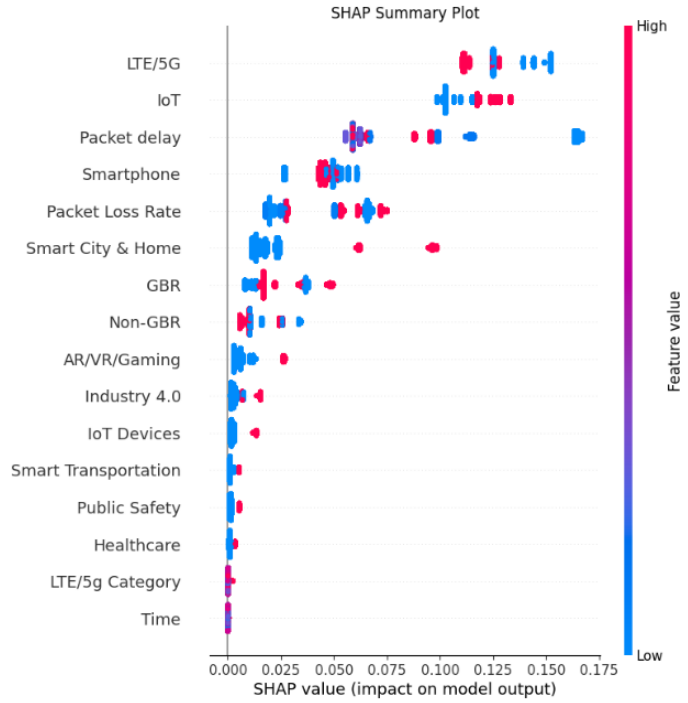


SHAP Bar Plot για το Slice Type 3

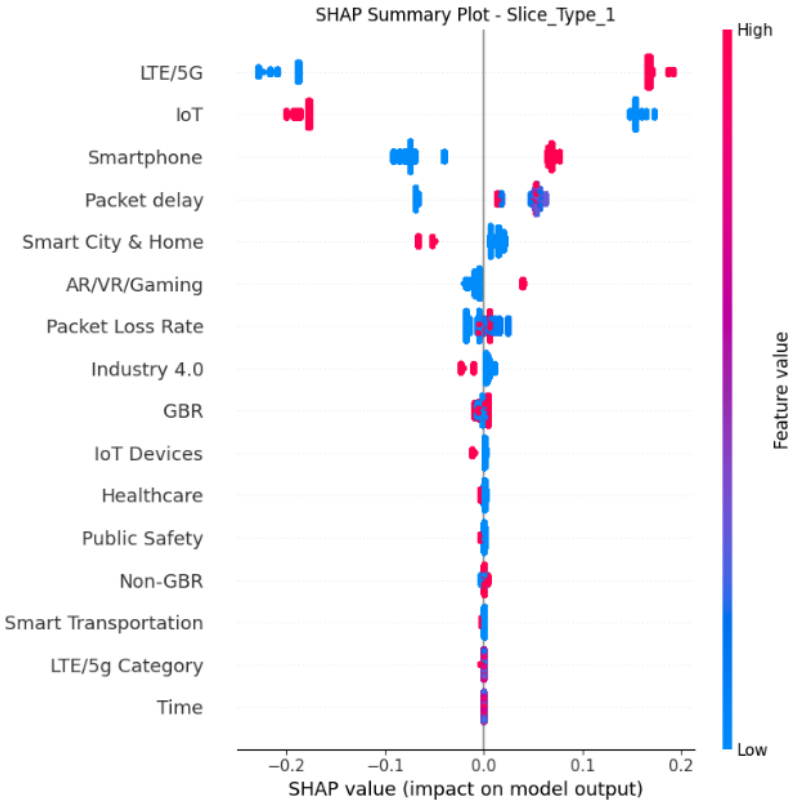
Στα παραπάνω bar plots, φαίνεται πόσο σημαντικό είναι κάθε feature για τα labels. Στο γενικό bar plot, συνυπολογίζονται όλα τα features και δείχνουν τη μέση τιμή των υπόλοιπων SHAP bar plots. Στα bar plots για τα slice type 1 και 2 , το LTE/5G επηρεάζει περισσότερο από κάθε άλλο feature το αποτέλεσμα του slice type ενώ το slice type 3 επηρεάζεται περισσότερο από το Packet delay.

```
# Calculate avg SHAP values for each slice
shap_values_avg = np.mean(np.abs(shap_values), axis=2)

# Plot SHAP values across all slices
plt.figure(figsize=(14, 7))
shap.summary_plot(shap_values_avg, features=features_scaled_df, show=False)
plt.title("SHAP Summary Plot")
plt.savefig("SHAP/Summary_plot/shap_summary_plot_avg.png", bbox_inches='tight')
plt.tight_layout()
plt.close()
```

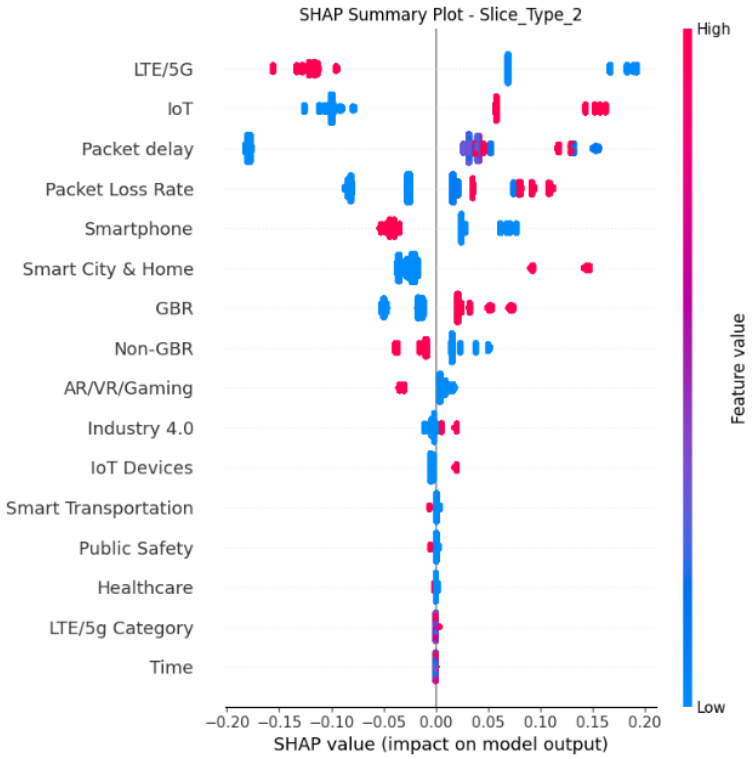
Στιγμιότυπο κώδικα του μέσου όρου όλων των SHAP values σε plot



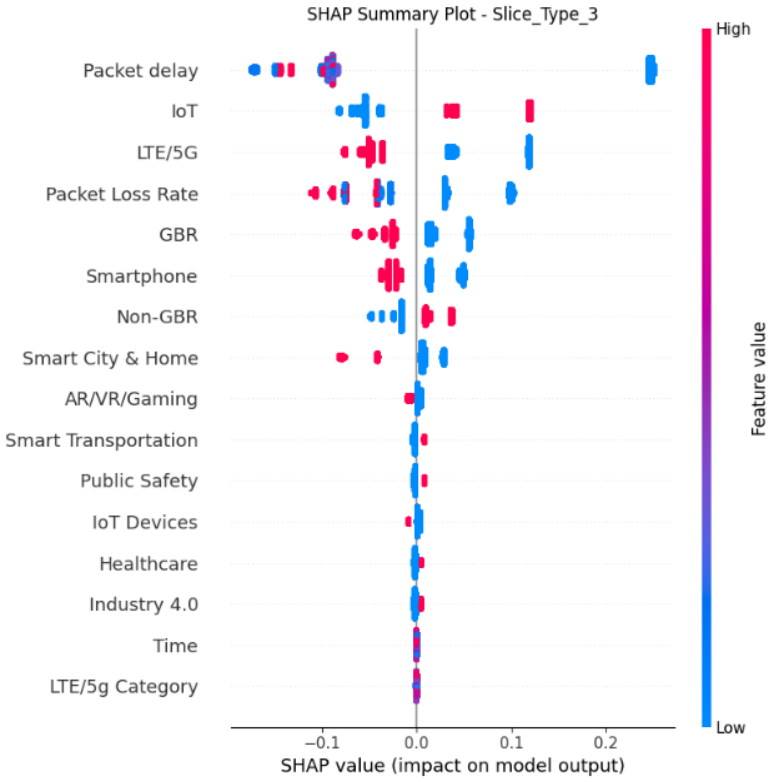
Summary Plot όλων των slice types



Summary plot του slice type 1



Summary plot του slice type 2



Summary plot του slice type 3

Τα παραπάνω plots περιέχουν τα feature καθώς και τα SHAP values, τα οποία δείχνουν τη τιμή του feature και τη βαρύτητα τους στο αποτέλεσμα του slice type αντίστοιχα. Για παράδειγμα, στο slice type 2 παρατηρείται πως για υψηλή τιμή LTE/5G(κόκκινο), το πακέτο δεν ανήκει στο συγκεκριμένο slice type, ενώ για χαμηλές τιμές(μπλε), το πακέτο ανήκει σε αυτό.

Feature Importances

Ταυτόχρονα, ελέγχθηκε και η ακρίβεια κατηγοριοποίησης με τη σταδιακή μείωση των features στο Random Forest. Τα αποτελέσματα δείχνουν πως με τα 3 πιο σημαντικά features(LTE/5G,IoT,Packet delay) το μοντέλο προβλέπει με 100% ακρίβεια το slice type που ανήκει κάθε πακέτο, ενώ με 2 features(LTE/5G,IoT), η ακρίβεια πέφτει στο 76,6%.

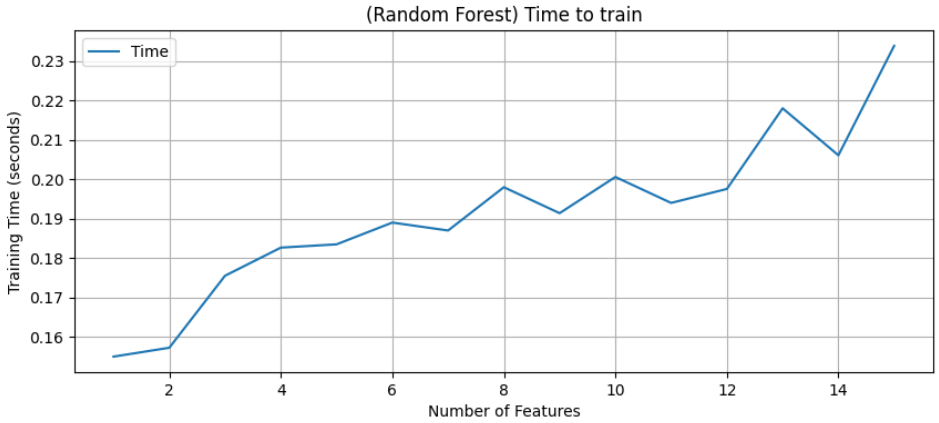
```
# Generate feature importances
importances = model.feature_importances_
forest_importances = pd.Series(importances, index=features.columns)
```

```
==============================

LTE/5G                  0.124695
IoT                     0.113108
Packet delay            0.094433
Smartphone              0.046823
Packet Loss Rate        0.041396
Smart City & Home       0.024527
GBR                     0.022483
Non-GBR                 0.015441
AR/VR/Gaming            0.007672
Industry 4.0            0.003862
IoT Devices             0.002556
Smart Transportation    0.001452
Public Safety           0.001447
Healthcare              0.001054
LTE/5g Category         0.000183
Time                    0.000116
dtype: float64
==============================
```

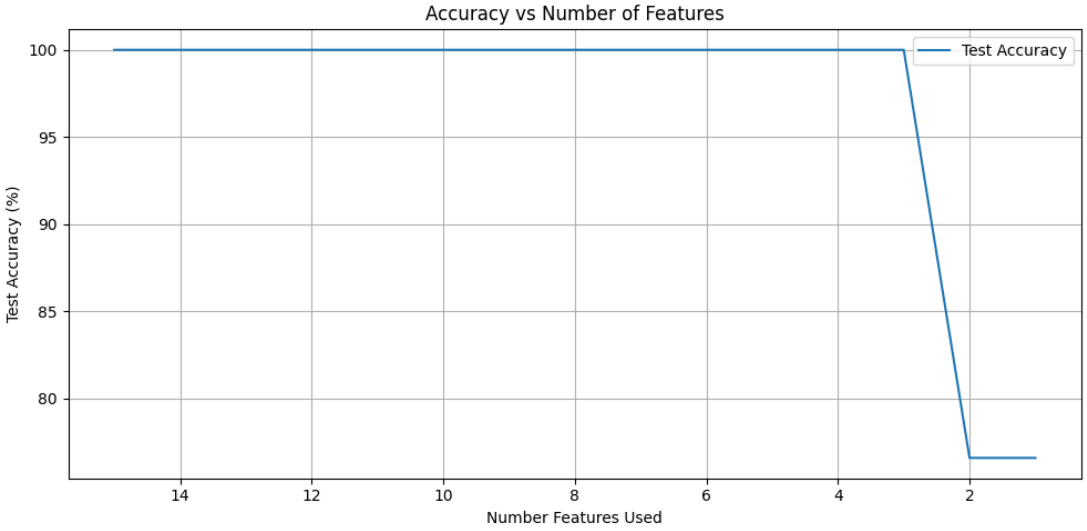
Feature Importances

```
-> Training model...

-> Train with 15 features, accuracy: 100.00%

-> Train with 14 features, accuracy: 100.00%

-> Train with 13 features, accuracy: 100.00%

-> Train with 12 features, accuracy: 100.00%

-> Train with 11 features, accuracy: 100.00%

-> Train with 10 features, accuracy: 100.00%

-> Train with 9 features, accuracy: 100.00%

-> Train with 8 features, accuracy: 100.00%

-> Train with 7 features, accuracy: 100.00%

-> Train with 6 features, accuracy: 100.00%

-> Train with 5 features, accuracy: 100.00%

-> Train with 4 features, accuracy: 100.00%

-> Train with 3 features, accuracy: 100.00%

-> Train with 2 features, accuracy: 76.60%

-> Train with 1 features, accuracy: 76.60%

Training done
```

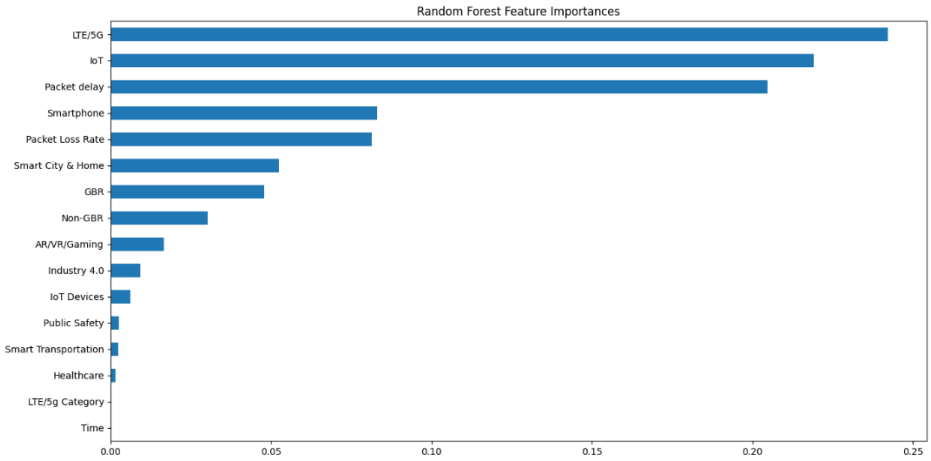
Εκπαίδευση με σταδιακή μείωση των features στο Random Forest Classifier



Χρόνος εκπαίδευσης με τη σταδιακή μείωση των features



Ακρίβεια μοντέλου με σταδιακή μείωση των features, από το λιγότερο προς το πιο σημαντικό feature του μοντέλου



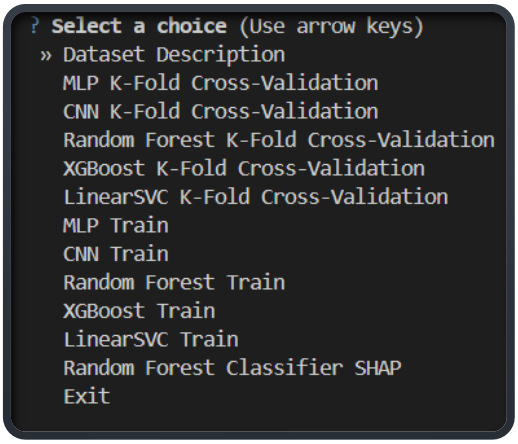
Bar Plot για τα Feature Importances του Random Forest Classifier

Παρατηρήσεις

Πακέτα Python που χρησιμοποιήθηκαν:

- pandas
- numpy
- matplotlib
- seaborn
- scikit-learn
- torch (PyTorch)
- xgboost
- joblib
- questionary
- time
- sys
- subprocess

Ο χρήστης μπορεί να επιλέξει ποιο αρχείο θα τρέξει μέσο του menu:



Για να τρέξουν οι κώδικες πρέπει να εκτελεστούν από το φάκελο «NetworkSlicingin6GNetworks». Ο οποίος περιέχει και τα επιπλέον plots και reports. Το αρχείο codes υπάρχει για επίδειξη του κώδικα και όχι εκτέλεσή του.

Πηγές

- Understanding important 5G concepts: What are eMBB, URLLC and mMTC? , https://www.verizon.com/about/news/5g-understanding-embb-urllc-mmtc
- Network slicing, https://www.techtarget.com/whatis/definition/network-slicing

- A Comprehensive Overview of Network Slicing for Improving the Energy Efficiency of Fifth-Generation Networks, https://www.mdpi.com/1424-8220/24/10/3242
- Multilayer Perceptrons in Machine Learning: A Comprehensive Guide, https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning
- Convolutional Neural Networks: A Comprehensive Guide, https://medium.com/thedeephub/convolutional-neural-networks-a-comprehensive-guide-5cc0b5eae175
- Random Forest Classification with Scikit-Learn, https://www.datacamp.com/tutorial/random-forests-classifier-python
- LinearSVC, https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
- XGBoost, https://www.geeksforgeeks.org/xgboost/