Εργασία Αναγνώριση Προτύπων και Μηχανική Μάθηση

Christos Alexopoulos (10618)

Panagiotis Koutris (10671)

Team 4

ΜΕΡΟΣ Α'

- Έχουμε 2 κλάσεις ω_1 (χωρίς στρες) και ω_2 (με στρες)
- Μας δίνεται ο δείκτης x που μπορεί να χρησιμοποιηθεί σε ένα σύστημα ταξινόμησης για να διαπιστωθεί κάθε φορά αν ο χρήστης αισθάνεται στρες ή όχι.
- Κατανομή πυκνότητας πιθανότητας: $p(x|\theta) = \frac{1}{\pi} \cdot \frac{1}{1+(x-\theta)^2}$
- Με βάση τις μετρήσεις των 12 συναδέλφων πήραμε τις εξής a-priori Πιθανότητες: $P(\omega_1) = \frac{7}{12}$, $P(\omega_2) = \frac{5}{12}$
- Καλούμαστε να υλοποιήσουμε έναν ταξινομητή μέγιστης πιθανοφάνειας

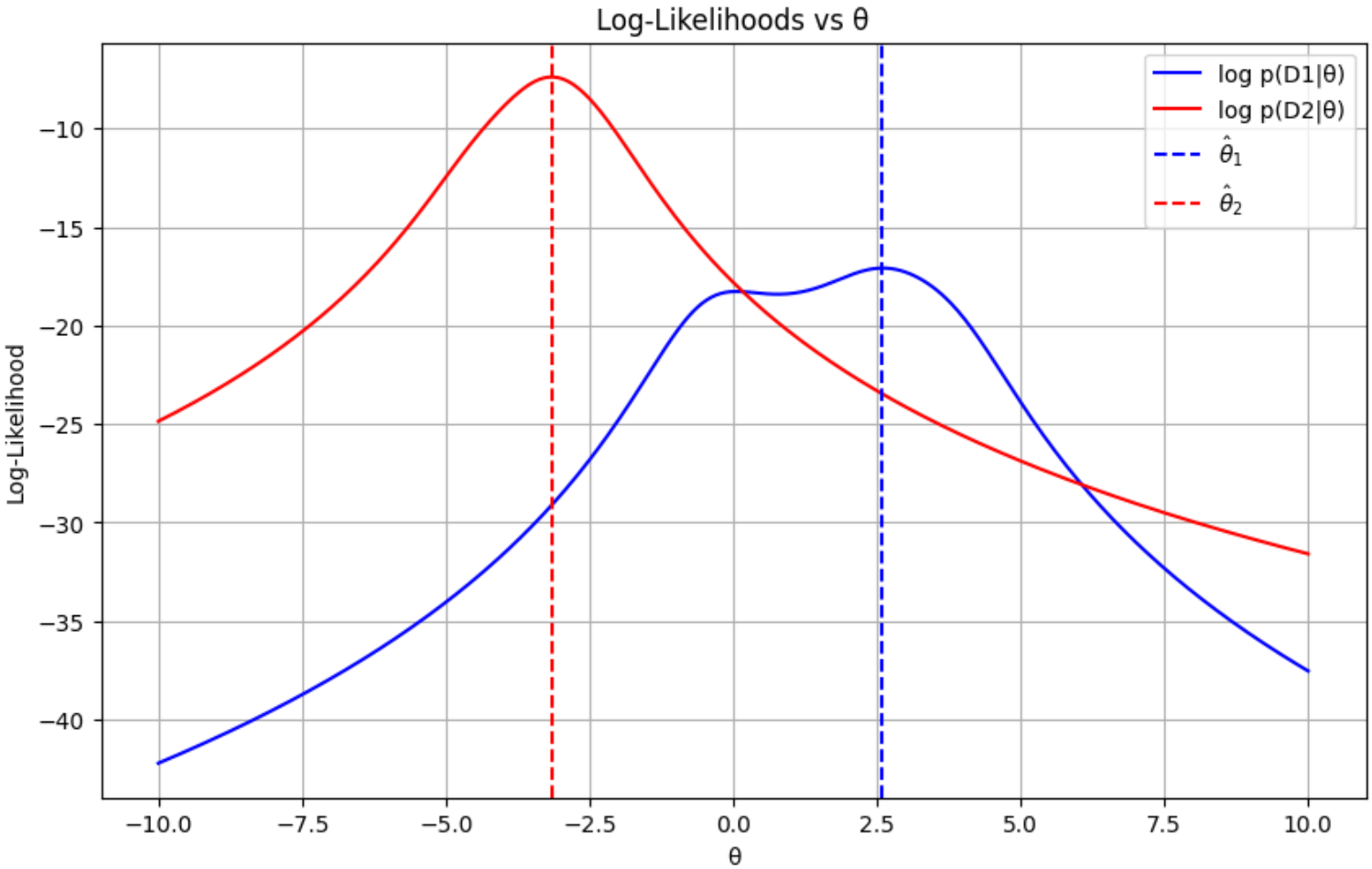
Ερώτημα 1^o

- Μας δίνονται τα εξής σύνολα τιμών:
- $D_1 = [2.8, 0.4, 0.8, 2.3, 0.3, 3.6, 4.1]$
- $D_2 = [4.5, 3.4, 3.1, 3.0, 2.3]$
- Ζητούμενα:
  - Εύρεση των βέλτιστων $\hat{\theta}_1$, $\hat{\theta}_2$
  - Απεικόνιση των $\log P(D_1|\theta)$, $\log P(D_2|\theta)$

Εύρεση $\hat{\theta_1}$, $\hat{\theta_2}$

- Λύνουμε την εξίσωση: $\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = 0$ **(1)**
- όπου: $l(\boldsymbol{\theta}) = \ln \prod_{n=1}^{N} p(x_n|\boldsymbol{\theta})$
- Για να προσεγγίσουμε τη λύση της εξίσωσης **(1)** χρησιμοποιούμε την αριθμητική μέθοδο argmax()
- Δοκιμάζουμε για το σύνολο τιμών $\theta = [-10, 10]$, βρίσκοντας με αυτό το τρόπο τη τιμή του βέλτιστου θ που μεγιστοποιεί τη πιθανότητα εμφάνισης των δεδομένων. Τα παρακάτω αποτελέσματα επιβεβαιώνονται και για άλλα σύνολα τμών θ στα οποία εμπεριέχεται η λύση.
- Αποτελέσματα:
  - $\hat{\theta_1} = 2.585$
  - $\hat{\theta_2} = -3.146$

Απεικόνιση $\log P(D_1|\theta)$, $\log P(D_2|\theta)$



Ερώτημα 2^o

- Χρησιμοποιώντας τη συνάρτηση διάκρισης:

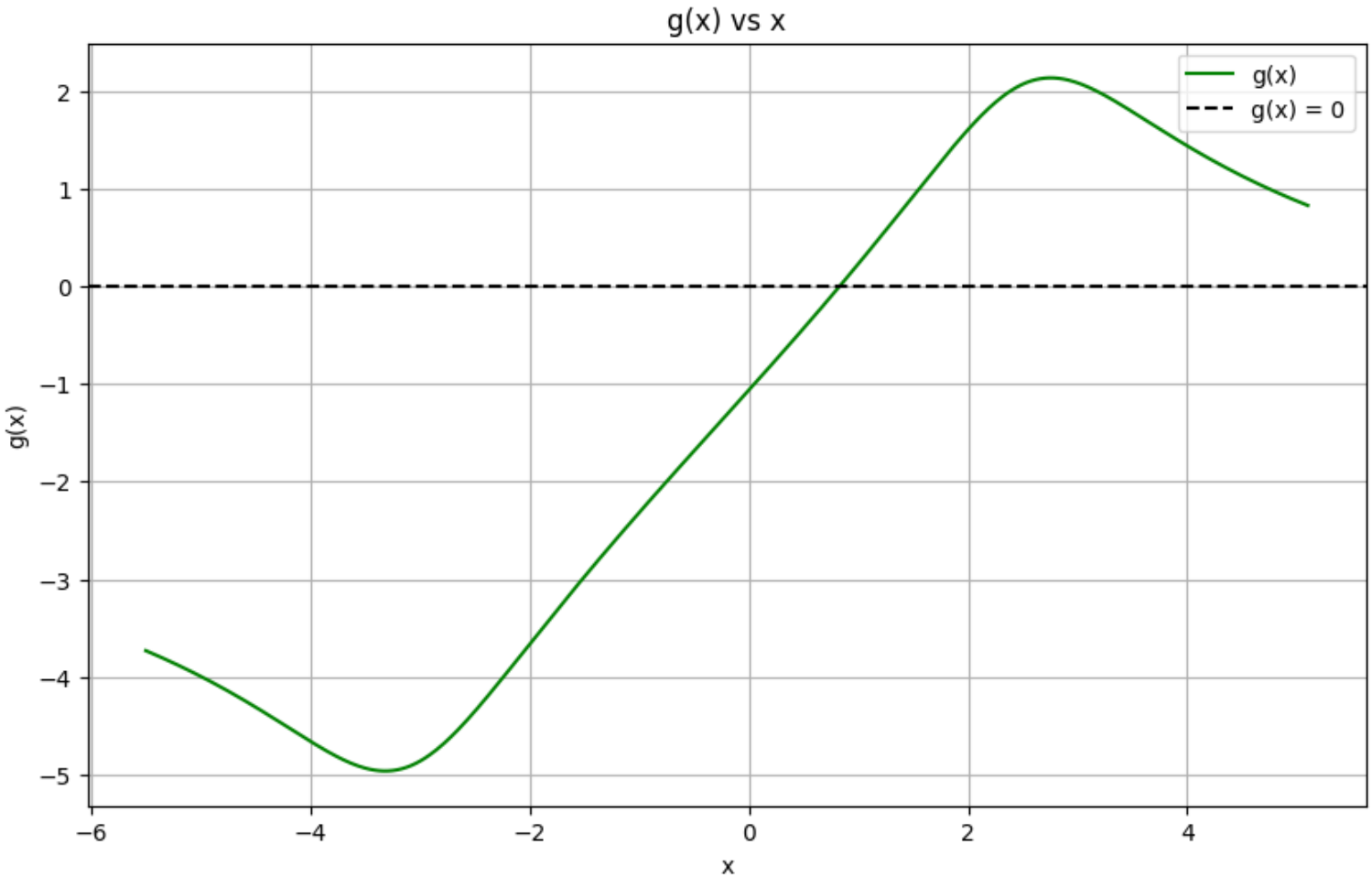
  $$g(x) = \log P(x|\hat{\theta}_1) - \log P(x|\hat{\theta}_2) + \log P(\omega_1) - \log P(\omega_2),$$

  ταξινομούμε τα σύνολα τιμών D_1, D_2

Κανόνας Απόφασης για Ταξινόμηση

- Ο κανόνας απόφασης που θα ακολουθήσουμε είναι ο εξής
  - $g(x) > 0 \rightarrow \omega_1$
  - $g(x) < 0 \rightarrow \omega_2$
- Με βάση αυτόν τον κανόνα η τιμή της $g(x)$ είναι:
  - θετική για τα δείγματα (2.8, 2.3, 3.6, 4.1)
  - αρνητική για όλα τα υπόλοιπα δείγματα
- Δηλαδή:
  - Τα δείγματα (2.8, 2.3, 3.6, 4.1) ανήκουν στην κλάση ω_1
  - τα υπόλοιπα ανήκουν στην ω_2
- Επομένως:
  - Ο ML ταξινομητής λειτουργεί καλά για τα δεδομένα της κλάσης ω_2 καθώς ταξινόμησε σωστά όλα τα δείγματα
  - Για τα δεδομένα της κλάσης ω_1, ο ταξινομητής παρουσίασε λάθη σε τρία δείγματα, τα (0.4, 0.8, 0.3).
- Παρακάτω παρατίθεται απεικόνιση της $g(x)$

Απεικόνιση $g(x)$



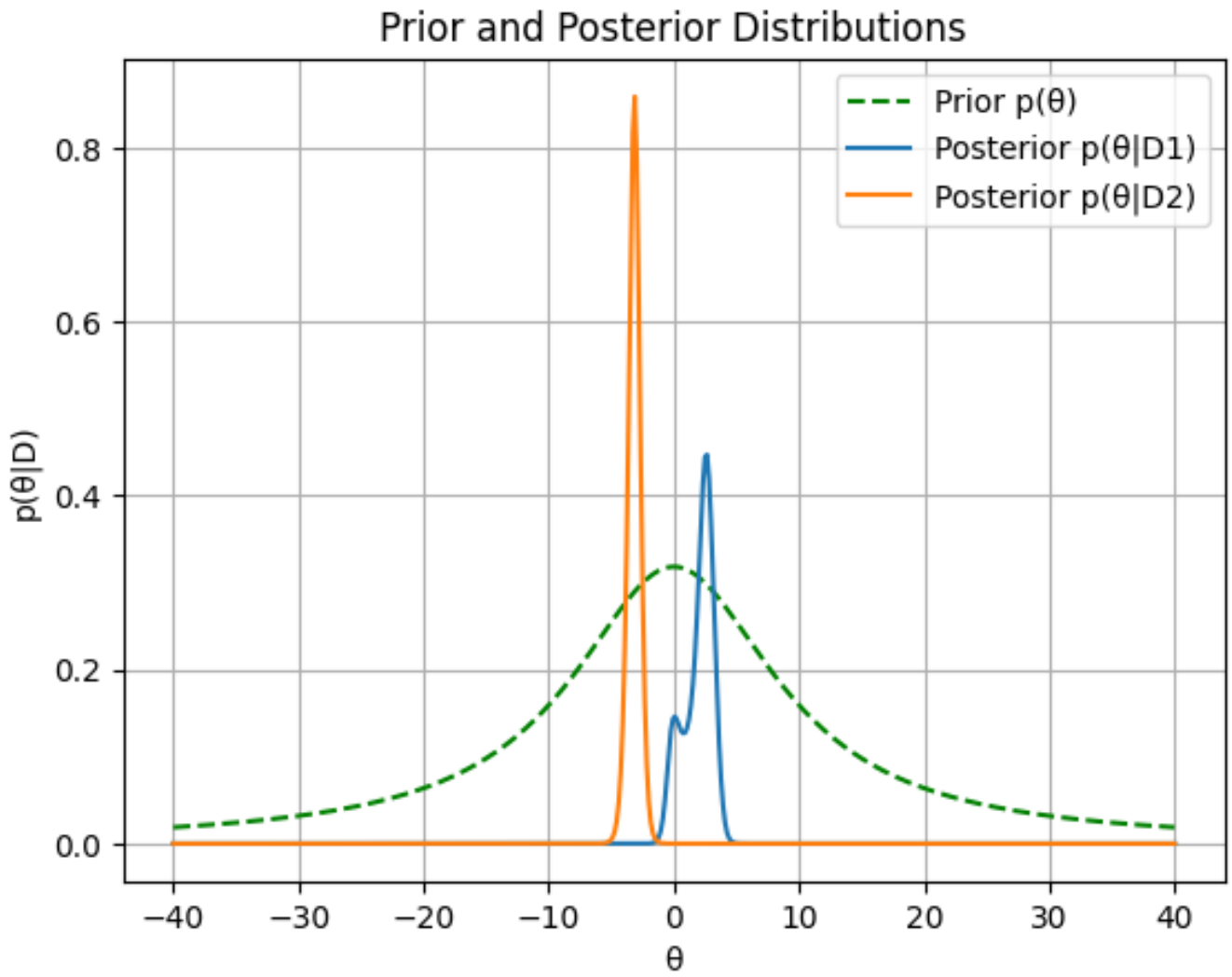
ΜΕΡΟΣ Β'

- Έχουμε την a-priori πιθανότητα $p(\theta) = \frac{1}{10\pi} \cdot \frac{1}{1+(\theta/10)^2}$
- Θα υπολογίσουμε:
  - την a posteriori πιθανότητα: $p(\theta|D)$
  - την πυκνότητα πιθανότητας $p(x|D_j), \;\; j = 1, 2$

Ερώτημα 1^o

- Ζητούμενο είναι η απεικόνιση των $p(\theta|D_j), j = 1, 2$
- Θα τα υπολογίσουμε ως εξής:
  - $p(D_j|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$
  - $p(\theta|D_j) = \frac{p(D_j|\theta)p(\theta)}{\int p(D_j|\theta)p(\theta)d\theta}$
- Παρακάτω παρατίθεται το σχεδιάγραμμα των $p(\theta|D_j), j = 1, 2$ από το οποίο παρατηρούμε ότι οι εκάστωτε καπλύλες είναι περισότερο συγκετρωμένες γύρω από τη περιοχή των δεδομένων σε σχέση με το $p(\theta)$
- Αυτό το αποτέλεσμα είναι αναμενόμενο καθώς οι posterior πιθανότητες λαμβάνουν υπόψη τόσο τα δεδομένα όσο και την prior $p(\theta)$

Απεικόνιση $p(\theta|D_1), p(\theta|D_2)$



Ερώτημα 2^o

- Εχουμε τη συνάρτηση διάκρισης:

$$h(x) = \log P(x|D_1) - \log P(x|D_2) + \log P(\omega_1) - \log P(\omega_2),$$

και υπολογίζουμε τις τιμές της για τα σύνολα δεδομένων D_1, D_2.

Αποτελέσματα

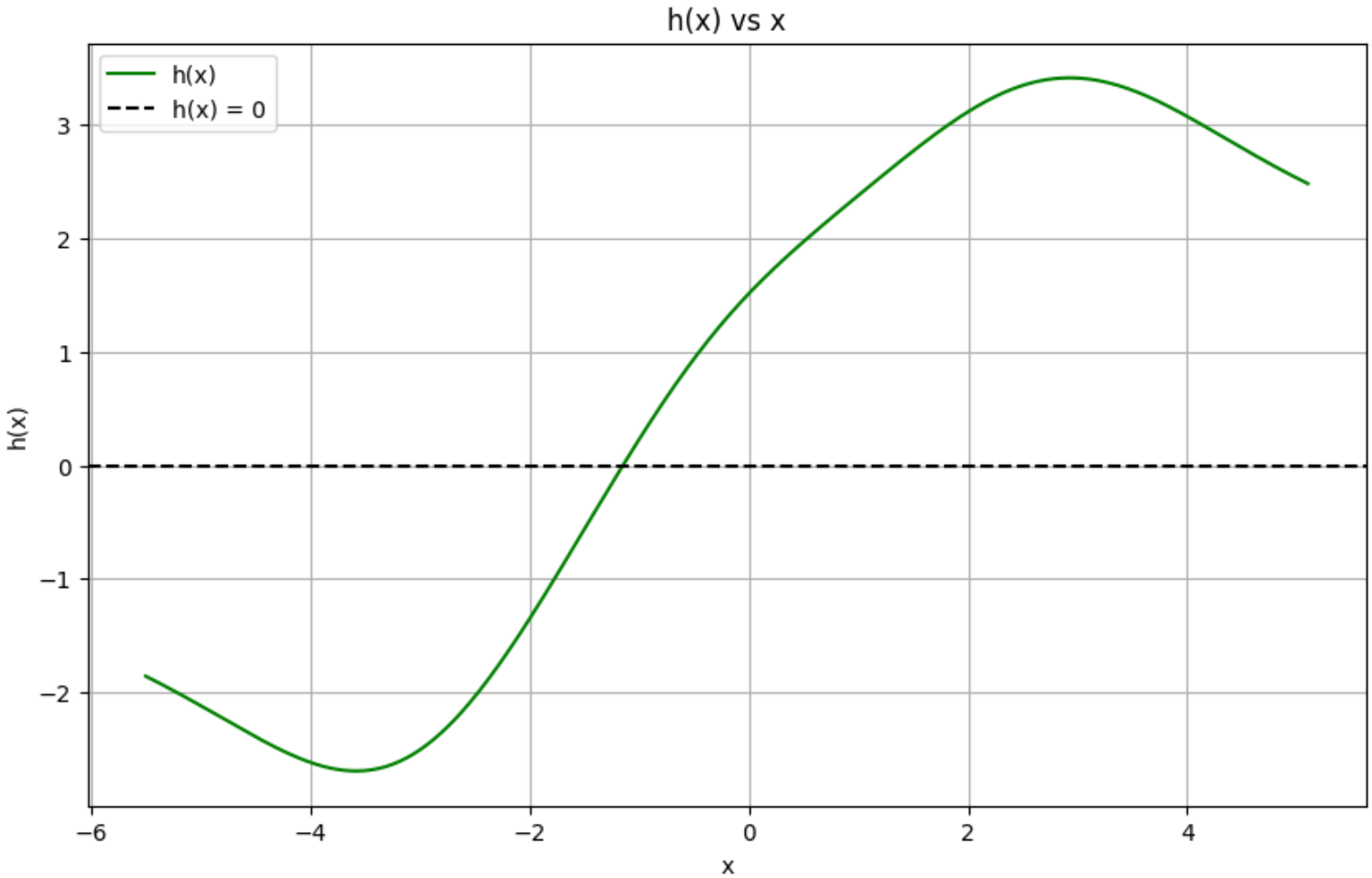
x	$h(x)$
2.8	3.410
-0.4	1.065
-0.8	0.527
2.3	3.274
-0.3	1.186
3.6	3.268
4.1	3.022
-4.5	-2.391
-3.4	-2.676
-3.1	-2.567
-3.0	-2.507
-2.3	-1.785

Table: Υπολογισμένες τιμές $h(x)$ για τις αντίστοιχες x

Κανόνας Απόφασης για Ταξινόμηση

- Παρατηρούμε ότι οι τιμές της $h(x)$ είναι θετικές για το σύνολο δεδομένων D_1 και αρνητικές για το D_2.
- Εχοντας υπόψη τον εξής κανόνα απόφασης:
  - $h(x) > 0 \rightarrow \omega_1$
  - $h(x) < 0 \rightarrow \omega_2$
- Παρατηρούμε ότι έχουμε τέλεια ταξινόμηση.
- Παρακάτω παρατίθεται απεικόνιση της $h(x)$

Απεικόνιση $h(x)$



Σύγκριση της μεθόδου εκτίμησης παραμέτρων κατά Bayes σε σχέση με τη μέθοδο της μέγιστης πιθανοφάνειας

- Στο συγκεκριμένο παράδειγμα παρατηρούμε ότι είναι καλύτερη η μέθοδος BE συγκριτικά με την ML.
- Η πρώτη έχει μηδενικό σφάλμα ταξινόμισης ενώ με τη δεύτερη ταξινομούνται εσφαλμένα 4 δείγματα.
- Αυτή η διαφορά στην απόδοση είναι αναμενόμενη καθώς η BE λαμβάνει υπόψη και την priori κατανομή του $p(\theta)$.
- Βέβαια, η ML είναι πιο απλή και ευκολότερη στην ερμηνεία και σε κάποια άλλη περίπτωση ενδεχομένως να είχαμε παρόποια απόδοση με την BE.

ΜΕΡΟΣ Γ'

- Σε αυτό το μέρος θα υλοποιήσουμε:
  - Έναν Decision Tree Ταξινομητή
  - Έναν Random Forest Ταξινομητή
- Για την εφαρμογή τους θα χρησιμοποιήσουμε από τη βιβλιοθήκη sklearn τους αλγορίθμους:
  - DesicionTreeClassifier
  - RandomForestClassifier

Ερώτημα 1^o

- Θα ταξινομήσουμε το 50% των δειγμάτων του dataset στις κλάσεις Iris setosa, Iris versicolor και Iris virginica, αφότου πρώτα εκπαιδεύσουμε τον ταξινομητή μας με το υπόλοιπο 50%.
- Για την ταξινόμηση θα αξιοποιήσουμε μόνο τα 2 πρώτα features των δειγμάτων, δηλαδή το μήκος και πλάτος των σεπάλων.
- Ζητούμενα:
  - Ακρίβεια ταξινόμησης
  - Βάθος δέντρου που δίνει την καλύτερη ακρίβεια ταξινόμησης
  - Απεικόνιση ορίων απόφασης για το παραπάνω βάθος

Υπολογισμός Ακρίβειας Ταξινόμησης

- Αρχικά φορτώνουμε το dataset και ορίζουμε τα training και testing set.
- Ορίζουμε τα πιθανά βάρη από 1 εώς 9
- Για κάθε ένα από τα βάθη ορίζουμε και εκπαιδεύουμε το Desicion Tree μοντέλο μας, υπολογίζοντας την ακρίβεια για καθένα από αυτά
- Κρίνεται απαραίτητο να αναφερθεί ότι για να αποφευχθεί η τυχαιότητα των αποτελεσμάτων σε κάθε τρέξιμο του κώδικα επιλέχθηκε σε όλα τα ερωτήματα αυτού του μέρους το random_state $= 42$

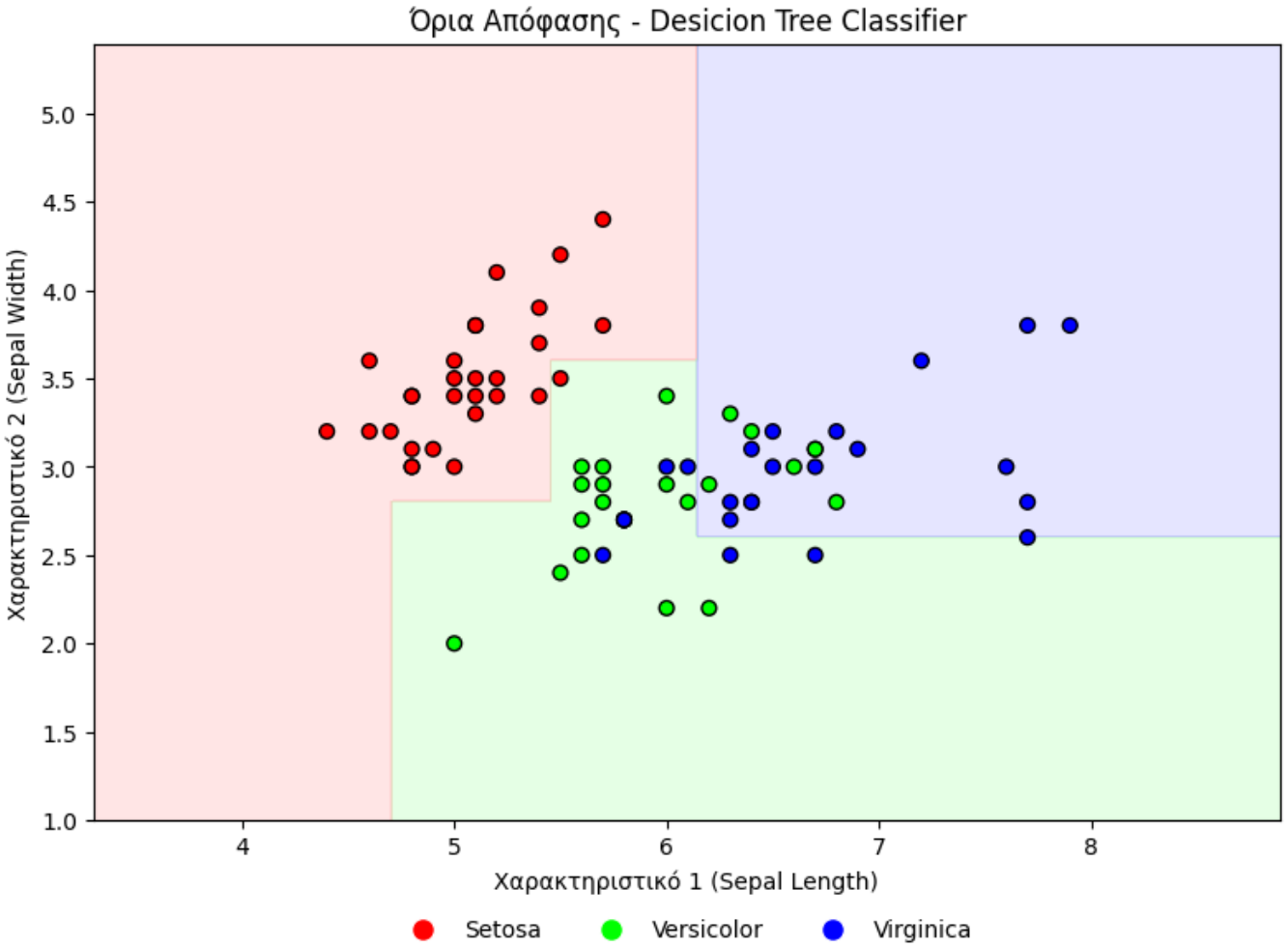
Αποτελέσματα

Βάθος	Ακρίβεια
1	0.640
2	0.773
3	**0.7866**
4	0.773
5	0.773
6	0.773
7	0.733
8	0.733
9	0.733

Table: Αποτελέσματα ακρίβειας για διαφορετικά βάθη ταξινομητή

- Στη συνέχεια θα απεικονίσουμε τα όρια απόφασης του ταξινομητή για το βάθος 3 με τη συνάρτηση contourf.

Όρια Απόφασης



Ερώτημα 2^o

- Θα υλοποιήσουμε έναν Random Forest ταξινομητή 100 δέντρων με την τεχνική Bootstrap.
- Θα χρησιμοποιήσουμε το $\gamma = 50\%$ του προηγουμένου συνόλου εκπαίδευσης για τη δημιουργία των συνόλων εκπαίδευσης του κάθε δέντρου.
- Το testing set είναι το ίδιο με το προηγούμενο ερώτημα
- Τα ζητούμενα και η προσέγγιση που θα ακολουθήσουμε θα είναι παρόμοια με το προηγούμενο ερώτημα

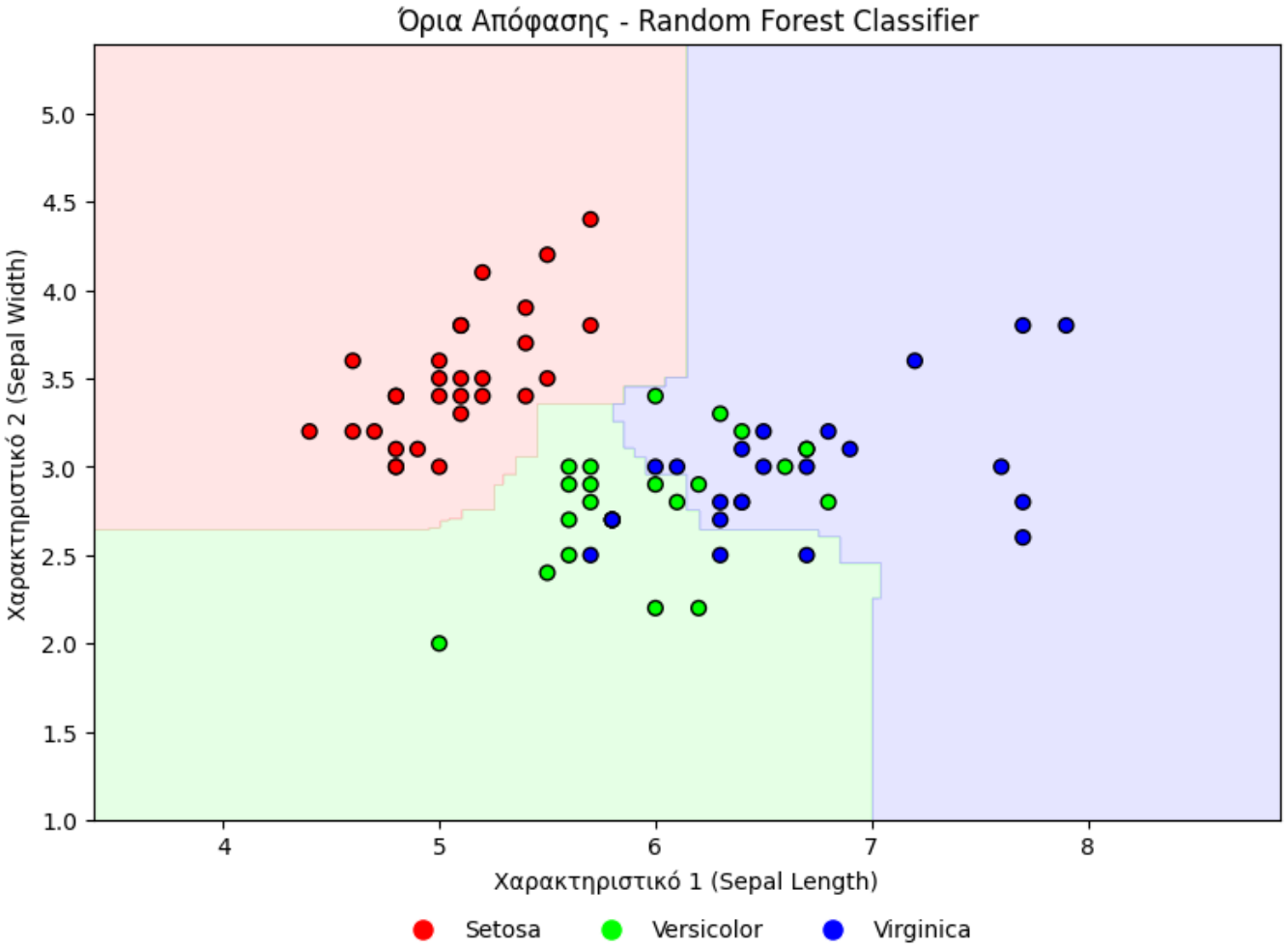
Ακρίβεια Ταξινόμησης

Βάθος	Ακρίβεια
1	0.76
2	**0.826**
3	0.8
4	0.8
5	0.786
6	0.786
7	0.786
8	0.773
9	0.786

Table: Αποτελέσματα ακρίβειας για διαφορετικά βάθη ταξινομητή.

- Στη συνέχεια θα απεικονίσουμε τα όρια απόφασης του ταξινομητή για το βάθος 2 με τη συνάρτηση contourf.

Όρια Απόφασης



Παρατηρήσεις (1/2)

- Ο RF ταξινομητής έχει καλύτερη ακρίβεια συγκριτικά με τον DT
- Αυτό το αποτέλεσμα αντικατοπτρίζεται και στα όρια απόφασης
  - Του DT είναι πιο ομαλά και γενικεύσιμα
  - Του RF είναι πιο σύνθετα και ειδικά
- Βέβαια, αν επικεντρωθούμε στη κλάση Iris Virginica, βλέπουμε ότι η ταξινόμηση είναι πολύ καλή και στις 2 περιπτώσεις σε αντίθεση με τις άλλες 2 κλάσεις
- Επίσης παρατηρήσαμε ότι με την επιλογή 2 άλλων features (π.χ. Μήκος Σεπάλου, Μήκος Πετάλου) είχαμε πολύ καλύτερο αποτέλεσμα ταξινόμησης ως προς όλες τις κλάσεις
- Δεν παρατηρούμε ιδιαίτερο overfiting με τον RF ταξινομητή, το οποίο οφείλεται στο μικρό βαθος δέντρων = 2

Παρατηρήσεις (2/2)

γ	Accuracy
0.1	0.8000
0.3	0.8133
0.5	0.8267
0.7	0.8267
1.0	0.8267

Table: Επίδραση του γ στην απόδοση

- Η απόδοση βελτιώνεται με την αύξηση του γ έως το $\gamma = 0.5$, όπου φτάνει τη μέγιστη ακρίβεια (82.67%). Μετά από αυτό, η απόδοση παραμένει σταθερή, υποδεικνύοντας ότι δεν απαιτείται η χρήση όλων των δεδομένων για κάθε δέντρο.

ΜΕΡΟΣ Δ'

- Σε αυτό το μέρος θα φτιάξουμε αλγορίθμους ταξινόμησης με διάφορες μεθόδους
- Training Set : datasetTV.csv
- Testing Set : datasetTest.csv
- Για κάθε μία από τις μεθόδους, θα υπολογίσουμε την ακρίβεια και θα επιλέξουμε το καλύτερο μοντέλο
- Από αυτό λοιπόν το μοντέλο θα εξάγουμε το διάνυσμα labelsX που θα αντιστοιχεί στις ετικέτες του testing set.

Μεθοδολογία

- **ΒΗΜΑ 1:** Grid Search με:
  - Cross-Validation

  για εύρεση βέλτιστων παραμέτρων
- **ΒΗΜΑ 2:** Εφαρμογή καλύτερων παραμέτρων με:
  - Training Split
- **ΒΗΜΑ 3:** Υπολογισμός Ακρίβειας

Σημείωση: Για κάθε μία από τις παρακάτω μεθόδους θα παρατίθεται αντίστοιχος κώδικας στο παράρτημα

ΜΕΘΟΔΟΣ 1: Gradient Boosting Classifier

- **Grid Search with Cross Validation**
- **Parameter Grid** :
  - $n\_estimators$: $[50, 100, 200]$
  - $learning\_rate$: $[0.05, 0.1, 0.2]$
  - max_depth : $[3, 4, 5]$
  - $subsample$: $[0.8, 0.9, 1.0]$
- **Best Parameters:**
  - $learning\_rate$: 0.2
  - max_depth : 5
  - $n\_estimators$: 200
  - $subsample$: 1.0
- **Accuracy:** 0.8326

ΜΕΘΟΔΟΣ 2: Random Forest Classifier

- **Grid Search with Cross Validation**
- **Parameter Grid:**
  - $n\_estimators : [100, 200, 300]$
  - $max\_depth : [10, 20, 30, None]$
  - $min\_samples\_split : [2, 5, 10]$
  - $min\_samples\_leaf : [1, 2, 4]$
- **Best Parameters**:
  - $max\_depth : 20$
  - $min\_samples\_leaf : 1$
  - $min\_samples\_split : 2$
  - $n\_estimators : 300$
- **Accuracy:** 0.8171

ΜΕΘΟΔΟΣ 3: MLP Classifier (Neural Network)

- **Grid Search with Cross Validation**
- **Parameter Grid:**
  - $hidden\_layer\_sizes : [(50,), (100,), (100, 50), (200, 100)]$
  - $activation : ['relu', 'tanh']$
  - $solver : ['adam', 'sgd']$
  - $alpha : [0.0001, 0.001, 0.01]$
  - $learning\_rate : ['constant', 'adaptive']$
- **Best Parameters:**
  - $activation :' tanh'$
  - $alpha : 0.01$
  - $hidden\_layer\_sizes : (200, 100)$
  - $learning\_rate :' constant'$
  - $solver :' adam'$
- **Accuracy:** 0.8542

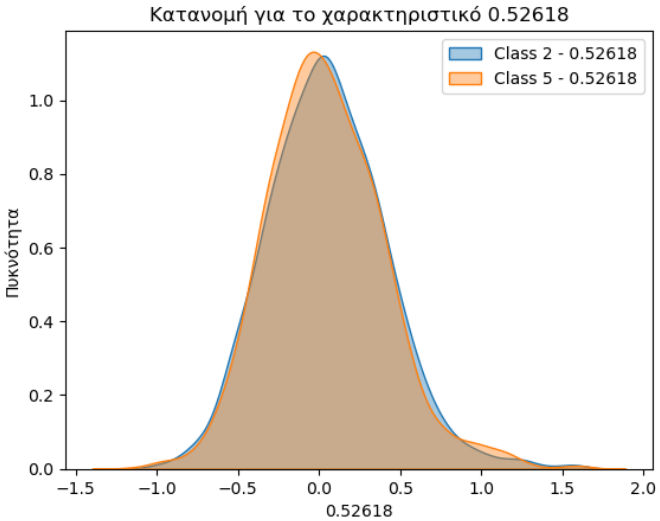
ΜΕΘΟΔΟΣ 4: SVM Classifier

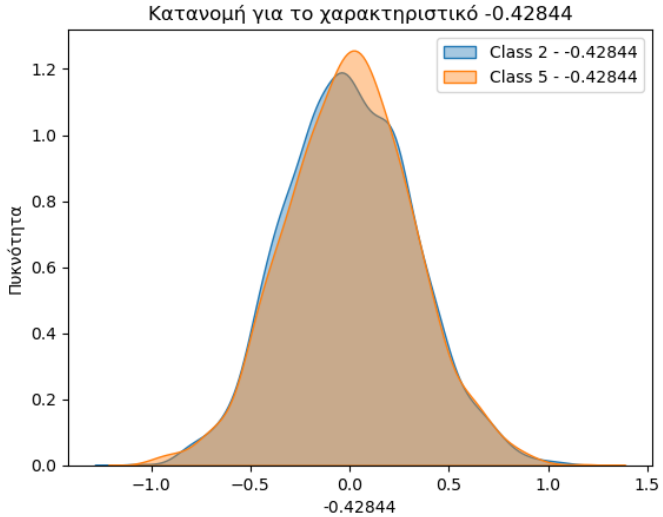
- **Grid Search with Cross-Validation for multiple kernels**
- **Parameter Grid:**
  - $C : [0.1, 1, 10, 50, 100, 200]$
  - $gamma : [0.001, 0.01, 0.1, 0.5, 1]$
  - $kernel : ['linear', 'rbf', 'poly', 'sigmoid']$
- **Best Parameters**:
  - $C : 10$
  - $gamma : 0.01$
  - $kernel : 'poly'$
- **Accuracy:** 0.8658

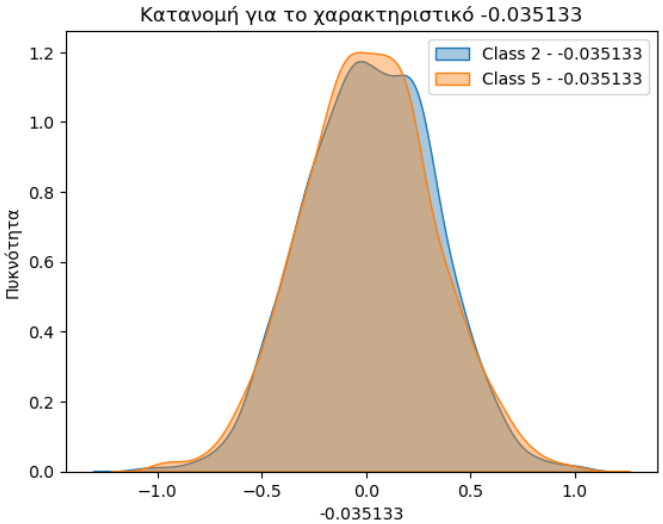
Παρατηρήσεις

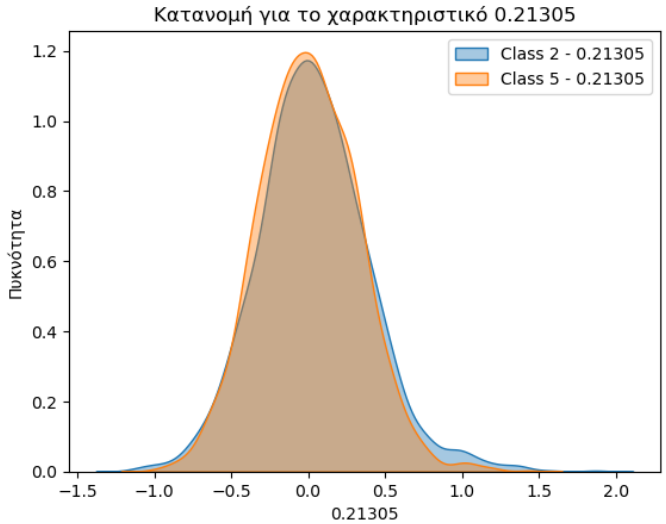
- Για κάθε μία από τις παραπάνω μεθόδους, εστιάσαμε στη **ταξινόμηση** της κάθε κλάσης.
- Σε κάθε υλοποίηση βγάλαμε το **Classification Report** και το **Confusion Matrix** για να δούμε το precision της κάθε κλάσης και ποίες κλάσεις μπερδεύει ο ταξινομητής
- Παρατηρήσαμε ότι:
  - Συστηματικά οι κλάσεις 2 και 5 έχουν το **μικρότερο Precision**
  - Υπάρχει **επικάληψη** των κλάσεων 2 και 5, δηλαδή αρκετά δείγματα της κλάσης 2 ταξινομούνται στην 5 και αντίστροφα.
- Αυτή η επικάλυψη φανερώνεται και στα παρακάτω διαγράμματα
- Κώδικας για όλα τα παραπάνω παρατίθεται στο παράρτημα

Κατανομή διάφορων ενδεικτικών χαρακτηριστικών









Απόπειρες Βελτίωσης

- **Χρήση συνδυασμού μοντέλων (Ιεραρχική Ταξινόμηση) + Χρήση PCA για Dimensionality Reduction**
  - Δεν παρατηρήθηκε ιδιαίτερη βελτίωση
- **Αλλαγή των class weights** δίνοντας μεγαλύτερη έμφαση στις κλάσεις 2 και 5 + **Χρήση PCA για Dimensionality Reduction**
  - Σε αυτή τη περίπτωση παρουσιάστηκε σημαντική βελτίωση

Τελικό Μοντέλο

- Το μοντέλο μας έχει ως εξής:
- PCA με:
  - $n\_components = 53$

  Κρατάμε τις 53 σημαντικότερες διαστάσεις, όπως προέκυψε με Cross-Validation
- SVM με τις εξής παραμέτρους:
  - 'kernel' = 'poly'
  - class 2 weight = 2
  - class 5 weight = 2

ΠΑΡΑΡΤΗΜΑ: Κώδικας Gradient Boosting Classifier

```python
# Gradient Boosting

# Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.ensemble import GradientBoostingClassifier

# Load data set from local path
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv")

# Get features X and labels y
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Initialize Parameter Grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'subsample': [0.8, 0.9, 1.0]
}

# GradientBoosting Initialization
model = GradientBoostingClassifier()

# KFold Cross-Validation with 5 folds
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Grid Search with Cross-Validation (n_jobs=-1 -> uses all available cores of the computer for faster results, verbose=2 -> prints progress updates)
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=kf, scoring='accuracy', n_jobs=-1, verbose=2)

# Train model with Grid Search
grid_search.fit(X, y)

# Print Results
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Cross-Validation Accuracy: {grid_search.best_score_:.4f}")

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

ΠΑΡΑΡΤΗΜΑ: Κώδικας Random Forest Classifier

```python
# Randomforest with Parameter Grid and Cross-Validation

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
import pandas as pd

# Load data from local path
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv")

# Get features X and labels y
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Initialize Randomforest
rf = RandomForestClassifier(random_state=42)

# Initialize Parameter Grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# GridSearchCV to get best parameters
grid_search = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search.fit(X, y)

# Print results
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

ΠΑΡΑΡΤΗΜΑ: Κώδικας MLP Classifier

```python
# MLP Classifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd

# Load dataset from local path
data = pd.read_csv('/Users/chris/Desktop/ML/project/datasetTV.csv', header = None)

# Get features X Labels y
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

print(f"Dataset shape: {X.shape}, {y.shape}")

# Test split, 80(training)-20(test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Initialize Parameter Grid
param_grid = {
    'hidden_layer_sizes': [(50,), (100,), (100, 50), (200, 100)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'sgd'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['constant', 'adaptive']
}

# Initialize MLP classifier
mlp = MLPClassifier(max_iter=300, random_state=42)

# GridSearchCV
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# Print best parameters
print("Best parameters found:", grid_search.best_params_)

# Print results
best_mlp = grid_search.best_estimator_
y_pred = best_mlp.predict(X_test)

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

ΠΑΡΑΡΤΗΜΑ: Κώδικας SVM Classifier

```python
# SVM with Cross Validation and multiple kernels
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
import pandas as pd

# Load dataset from local path
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv", header = None)

# Get features X and labels y
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Initialize SVM
svm = SVC()

# Initialize Parameter Grid (for multiple kernels)
param_grid = {
    'C': [0.1, 1, 10, 50, 100, 200],
    'gamma': [0.001, 0.01, 0.1, 0.5, 1],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid']
}

# GridSearchCV for getting best params
grid_search = GridSearchCV(svm, param_grid, cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search.fit(X, y)

# Print Results
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", grid_search.best_score_)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

ΠΑΡΑΡΤΗΜΑ : Κώδικας Διαγραμμάτων Επικάλυψης

```python
# Get the feature distribution for classes 2 and 5

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load data set from local path
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv")

# Get only classes 2 and 5
data_2_5 = data[data.iloc[:, -1].isin([2, 5])]

# Distribution for classes 2 and 5
class_2 = data_2_5[data_2_5.iloc[:, -1] == 2]
class_5 = data_2_5[data_2_5.iloc[:, -1] == 5]

# Average and standard deviation
print("\nΜέσες τιμές για Κλάση 2:")
print(class_2.mean())

print("\nΜέσες τιμές για Κλάση 5:")
print(class_5.mean())

print("\nΤυπικές αποκλίσεις για Κλάση 2:")
print(class_2.std())

print("\nΤυπικές αποκλίσεις για Κλάση 5:")
print(class_5.std())

# Visualization of classes distribution
plt.figure(figsize=(12, 8))
for column in data.columns[:-1]:
    sns.kdeplot(class_2[column], label=f'Class 2 - {column}', fill=True, alpha=0.4)
    sns.kdeplot(class_5[column], label=f'Class 5 - {column}', fill=True, alpha=0.4)
    plt.title(f"Κατανομή για το χαρακτηριστικό {column}")
    plt.xlabel(column)
    plt.ylabel("Πυκνότητα")
    plt.legend()
    plt.show()

# Scatter plot for feature pairs
plt.figure(figsize=(12, 8))
sns.scatterplot(data=class_2, x=data.columns[0], y=data.columns[1], label="Class 2", alpha=0.7)
sns.scatterplot(data=class_5, x=data.columns[0], y=data.columns[1], label="Class 5", alpha=0.7)
plt.title(f"Scatter plot για {data.columns[0]} και {data.columns[1]}")
plt.xlabel(data.columns[0])
plt.ylabel(data.columns[1])
plt.legend()
```

ΠΑΡΑΡΤΗΜΑ : Κώδικας Ιεραρχικής Ταξινόμησης

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import KernelPCA

# Load dataset from the CSV file
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv")

# Separate features (X) and labels (y) - last column is the label
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the first SVM for all 5 classes
model = SVC(kernel='poly', random_state=42, class_weight='balanced')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Select samples classified as class 2 or 5
class_2_5_indices = np.where((y_pred == 2) | (y_pred == 5))[0]
filtered_samples = X_test[class_2_5_indices]
filtered_labels = y_pred[class_2_5_indices]

# Apply Kernel PCA with 'poly' kernel on the filtered samples
kpca = KernelPCA(n_components=100, kernel='poly', random_state=42)
filtered_samples_kpca = kpca.fit_transform(filtered_samples)

# Train a second SVM for binary classification (classes 2 and 5)
filtered_labels_binary = np.where(filtered_labels == 2, 0, 1)  # Class 2 -> 0, Class 5 -> 1
model_2_5 = SVC(kernel='rbf', random_state=42)
model_2_5.fit(filtered_samples_kpca, filtered_labels_binary)

# Predict for the second SVM
y_pred_2_5 = model_2_5.predict(filtered_samples_kpca)

# Merge predictions: replace class 2 and 5 predictions with the second SVM's output
final_predictions = np.copy(y_pred)
final_predictions[class_2_5_indices] = np.where(y_pred_2_5 == 0, 2, 5)

# Calculate the final accuracy
final_accuracy = accuracy_score(y_test, final_predictions)
print(f'Final merged accuracy: {final_accuracy}')

# Print confusion matrix and classification report
print("Confusion Matrix:")
print(confusion_matrix(y_test, final_predictions))

print("Classification Report:")
print(classification_report(y_test, final_predictions))
```

ΠΑΡΑΡΤΗΜΑ : Κώδικας Τελικού Μοντέλου

```python
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
from sklearn.decomposition import PCA
import pandas as pd

# Load the dataset
data = pd.read_csv("/Users/chris/Desktop/ML/project/datasetTV.csv", header=None)
X = data.iloc[:, :-1].values  # Features (all columns except the last)
y = data.iloc[:, -1].values   # Labels (last column)

# Apply PCA for dimensionality reduction
pca = PCA(n_components=53)
X_pca = pca.fit_transform(X)

# Train SVM with Cross-Validation
svm = SVC(kernel='poly', class_weight={2: 2, 5: 2}, random_state=42)

# Perform 5-fold cross-validation
scores = cross_val_score(svm, X_pca, y, cv=5)

# Print accuracy for each fold and the average accuracy
print(f"Cross-validation accuracy for each fold: {scores}")
print(f"Mean accuracy: {scores.mean():.4f}")
```