פרויקט סיום - למידת מכונה

בן דבוש ת.ז. 316014760
נעמה שפוני ת.ז 318607314

תקציר:

הפרויקט משתמש בשלושה Datasets שונים TESS,SAVEE,RAVDESS
כל אחד מכילים קטעי אודיו קצרים של אנשים כך שכל קטע אודיו מתויג לפי אחד מהרגשות- כעס, גועל, פחד, אושר, הפתעה נעימה, עצב. המטרה שלנו היא ליצור מודל שיכול לזהות את הרגשות של האדם לפי האינטואיציה בקטע אודיו בלי תלות בטקסט שנאמר.
קישור לגיט

שאלת מחקר:

בהינתן קטע אודיו של נקבה או זכר האם ניתן לזהות את הרגש שלהם לפי אינטונצית הדיבור?

המאגרים:

מאגר 1:

Toronto emotional speech set (TESS)
https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess
מערך הוא קטעי אודיו של נקבה בלבד
2800 קבצי אודיו
כל הקלטה מתויגת לפי רגש: כעס, גועל, פחד, אושר, הפתעה נעימה, עצב ונייטרלי

מאגר 2:

Surrey Audio-Visual Expressed Emotion (SAVEE)
https://www.kaggle.com/datasets/ejlok1/surrey-audiovisual-expressed-emotion-savee
מערך הוא קטעי אודיו של זכר בלבד
480 קבצי אודיו
כל הקלטה מתויגת לפי רגש: כעס, גועל, פחד, אושר, הפתעה נעימה ועצב.

מאגר 3:

RAVDESS Emotional speech audio
https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio
מערך מכיל 24 שחקנים מקצועיים (12 נשים, 12 גברים)
1440 קבצי אודיו
כל הקלטה מתויגת לפי רגש: כעס, גועל, פחד, אושר, הפתעה נעימה ועצב.

סיכום של האלגוריתמים:

הכנת הדאטה:

בפרויקט משתמש בשלושה מערכי נתונים שונים: TESS, SAVEE ו-RAVDESS. אנחנו משתמשים ב-3 פונקציות שונות ומותאמות אישית לכל אחד ממערכי הנתונים כדי לחלץ נתיבי קבצים ורגשות תואמים, וליצור DataFrames משותף ובעל שפה אחידה.
הפיכת נתוני אודיו גולמיים לייצוג מספרים משמעותי אנחנו משתמשים ספריית Librosa, עם 5 תכונות אודיו

1. **Zero Crossing Rate**: זוהי מדידה של מספר הפעמים שבהן אות אודיו חוצה את ציר ה-X (הקו האופקי) בתוך יחידת זמן. ZCR משמשת למדידת הראשיות של האודיו. ערכים גבוהים של ZCR מעידים על אודיו עם יותר רעש (כמו פיצוצים או קליקים), בעוד שערכים נמוכים מעידים על אודיו חלק ושקט יותר.
2. **Chroma Short-Time Fourier Transform**: זהו ייצוג של האודיו שמבצע המרה מהזמן לפקד של תדרים, ומנטר את התדרים לפי התכנים המוזיקליים שלהם. ה-Chroma STFT מתמקד בניתוח צבעי התדרים של האודיו בכל חלון זמן קצר ומספק תובנות על המוזיקה וההרמוניה באודיו.
3. **Mel-Frequency Cepstral Coefficients**: זהו ייצוג של תכני אודיו המשתמש בממדי התדר על פי סקאלה שמחקה את הדרך שבה אוזן האדם שומעת. MFCCים הם תכונות שמייצגות את האודיו בצורה של מקטעים טמפורליים ויכולים לשמש לצורך זיהוי דיבור, זיהוי רגשות, ועוד.
4. **Root Mean Square**: זוהי מדידה של עוצמת האות האודיו. RMS משמשת למדידת האנרגיה של האודיו על פני זמן, והיא משקפת את עוצמת הקול של האודיו.
5. **Mel Spectrogram**: זהו ייצוג של תדרים אודיו במונחים של סקאלה מלית, כלומר סקאלה שמחקה את האופן שבו האוזן האנושית שומעת תדרים. הוא מציג את התדרים של האודיו על פני זמן ומאפשר ניתוח ויזואלי של תכני האודיו לפי סקאלה מלית.

כדי לשפר את היכולות של המודל, מיושמות טכניקות הגדלת נתונים כגון **מתיחת זמן והסטת גובה**.
לאחר יצירת הפיצ'רים כתבנו הרצות ל-5 סוגי אלגוריתמים:
KNN, Decision Tree, Random Forest, Support Vector Machine,Neural Network

K-Nearest Neighbors:

הקוד שבניינו ל-KNN מכיל מספר פרמטרים חשובים. הראשון הוא n_neighbors, המגדיר את מספר השכנים הקרובים שיש לקחת בחשבון, עם ערכים מותאמים אישית (3, 5, 7, 9, 13, 17). test_size בפונקציה train_test_split קובע כי 20% מהנתונים ישמשו לסט הבדיקה, ו-random_state מבטיח חלוקה אקראית עקבית. n_runs מייצג את מספר ההרצות הכוללות (20), והשלב הזה מאפשר לבדוק את המודל על פני חלוקות שונות של הנתונים. עבור כל ערך של k, מתבצעות מספר חלוקות אקראיות שונות, כאשר בכל חלוקה המודל מוערך מחדש. הדיוקים מההרצות השונות נאספים ומחושבים ממוצע, כך שהממוצע מספק הערכה מדויקת יותר של ביצועי המודל ומפחית את השפעת המקריות.

Decision Tree:

הקוד שבנינו מיישם את האלגוריתם של Decision Tree, שמבצע הערכה של המודל על ידי הרצה של האלגוריתם מספר פעמים עם פרמטרים שונים. הקוד מריץ את המודל 20 פעמים עבור כל סט פרמטרים, כאשר בכל הרצה הדאטה מתחלק ל-80% נתוני אימון ו-20% נתוני בדיקה.
הפרמטרים השונים שמוגדרים ב-parameters הם:

- max_depth: העומק המקסימלי של עץ ההחלטות. ערכים אפשריים יכולים להיות מספר שלם כמו 3, 5, 7, או None עבור עומק בלתי מוגבל.
- criterion: הקריטריון למדידת איכות הפיצול. יכול להיות gini או entropy. ההבדל ביניהם הוא:
  - Gini: מדד טוהר שמעדיף צמתים עם חלוקה לא אחידה יותר. ככל שערך ה-Gini נמוך יותר, הצומת טהור יותר.

- Entropy: מדד אי-סדר המבוסס על תורת המידע. ככל שה-Entropy נמוכה יותר, הצומת טהור יותר. המדד הזה נותן יותר משקל לחלוקה שווה של הדוגמאות.
- min_samples_split: המספר המינימלי של דוגמאות הדרושות כדי לפצל צומת.
- min_samples_leaf: המספר המינימלי של דוגמאות הדרושות בכל עלה של העץ.

Random Forest:

הקוד שבנינו מיישם את האלגוריתם של Random Forest, שמבצע הערכה של המודל על ידי הרצה של האלגוריתם מספר פעמים עם פרמטרים שונים. הקוד מריץ את המודל 20 פעמים עבור כל סט פרמטרים, כאשר בכל הרצה הדאטה מתחלק ל-80% נתוני אימון ו-20% נתוני בדיקה.
הפרמטרים השונים שמוגדרים ב-parameters הם:

- n_estimators: מספר העצים במודל. ערכים אפשריים יכולים להיות מספרים שלמים כמו 100 או 150.
- max_depth: העומק המקסימלי של כל עץ. אפשר לקבוע ערך כמו 3, 5, 7, או None עבור עומק בלתי מוגבל.
- criterion: הקריטריון למדידת איכות הפיצול. יכול להיות gini או entropy. ההבדל ביניהם הוא:
  - Gini: מדד טוהר המעדיף צמתים עם חלוקה לא אחידה יותר. ככל שערך ה-Gini נמוך יותר, הצומת טהור יותר.
  - Entropy: מדד אי-סדר המבוסס על תורת המידע. ככל שה-Entropy נמוכה יותר, הצומת טהור יותר. המדד הזה נותן יותר משקל לחלוקה שווה של הדוגמאות.
- min_samples_split: המספר המינימלי של דוגמאות הדרושות כדי לפצל צומת.
- min_samples_leaf: המספר המינימלי של דוגמאות הדרושות בכל עלה של העץ.

באמצעות הגדרות אלו, הקוד מספק מסגרת להערכת ביצועי המודל ולבחינת השפעת הפרמטרים השונים על תוצאותיו.

Support Vector Machine:

הקוד שבנינו מיישם את האלגוריתם של SVM (Support Vector Machine), שמבצע הערכה של המודל על ידי הרצה של האלגוריתם מספר פעמים עם פרמטרים שונים. הקוד מריץ את המודל 20 פעמים עבור כל סט פרמטרים, כאשר בכל הרצה הדאטה מתחלק ל-80% נתוני אימון ו-20% נתוני בדיקה.
הפרמטרים השונים שמוגדרים ב-configurations הם:

- C: משקל הרגולציה של המודל. ערך גבוה (כמו 1.0) גורם למודל להיות פחות רגולרי, בעוד שערך נמוך יכול למנוע התחברות (overfitting).
- kernel: סוג הפונקציה הקרנלית שמשמשת את המודל. האפשרויות הן:
  - 'linear': מתאימה כאשר הנתונים ניתנים להפרדה בקו ישר.
  - 'rbf': מתאימה לנתונים שאינם ליניאריים.
  - 'poly': מתאימה כאשר קיימת תלות פולינומית בין התכונות.
  - 'sigmoid': פונקציה סיגמואידית, פחות נפוצה בשימוש בהשוואה לפונקציות אחרות.

Neural Network:

neural network מסוג EmotionNN עם hidden layer אחת . השתמש בפונקציית ReLU לפלט.
חילקנו את הדאטסט שלנו ל80% אימון 20% טסט, פונקציית loss אנחנו משתמשים בCrossEntropyLoss והאופטימיזטור Adam. אימנו את המודל במשך 50 אפוקים, תוך עדכון המשקלים כדי למזער את loss .

תוצאות:

- knn model

```
Evaluating file: features.csv

Overall Results:
File: features.csv
k=3: 0.86
k=5: 0.85
k=7: 0.84
k=9: 0.83
k=13: 0.82
k=17: 0.81
```

decision tree model-

```
Evaluating file: features.csv

Overall Results:
File: features.csv
{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.35
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.44
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.66
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.53
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.64
```

Random Forests-

```
Overall Results:
File: features.csv
{'n_estimators': 100, 'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.52
{'n_estimators': 150, 'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.64
{'n_estimators': 100, 'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.80
{'n_estimators': 150, 'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.80
{'n_estimators': 100, 'max_depth': 3, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.53
{'n_estimators': 150, 'max_depth': 7, 'criterion': 'entropy', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.72
(.venv) PS C:\Users\Ben\Desktop\ML_\ml_speech_emotion_recognition>
```

Support Vector Machine model-

```
Evaluating file: features.csv

Overall Results:
File: features.csv
C=1.0, kernel=linear: 0.71
C=1.0, kernel=rbf: 0.76
C=1.0, kernel=poly: 0.45
C=1.0, kernel=sigmoid: 0.56
```

Neural Networks model -

```
_algorithms/nn.py
Epoch [10/50], Loss: 1.0776
Epoch [20/50], Loss: 0.2241
Epoch [30/50], Loss: 0.4563
Epoch [40/50], Loss: 0.2693
Epoch [50/50], Loss: 0.1016
Test Accuracy: 84.06%
```

ניסיון לשיפור תוצאות עם PCA

ערכנו ניסוי המשלב PCA במסווג ה-KNN שלנו כדי לקבוע אם הפחתת הממדים של מערך הנתונים שלנו יכולה לשפר את ביצועי המודל. על ידי שמירה על רמות שונות (95%, 90%, 85% ו-80%), שאפנו לפשט את הנתונים, לשפר את הדיוק והיעילות החישובית. עם זאת, התוצאות הראו שככל שעשינו פחות שונות, הדיוק ירד, מה שמצביע על כך שמידע משמעותי אבד. זה מדגיש את הסיכון של צמצום יתר של הממדיות, מה שעלול להוביל לאובדן נתונים קריטיים ובסופו של דבר לפגוע בביצועי המודל.

```
Overall Results:
File: csvResults/features.csv
PCA with 95.0% variance retained
k=3: 0.86
k=5: 0.85
k=7: 0.84
k=9: 0.83
k=13: 0.82
k=17: 0.81
PCA with 90.0% variance retained
k=3: 0.83
k=5: 0.83
k=7: 0.82
k=9: 0.81
k=13: 0.80
k=17: 0.79
PCA with 85.0% variance retained
k=3: 0.81
k=5: 0.81
k=7: 0.80
k=9: 0.80
k=13: 0.79
k=17: 0.78
PCA with 80.0% variance retained
k=3: 0.79
k=5: 0.79
k=7: 0.79
k=9: 0.78
k=13: 0.77
k=17: 0.77
```

ערכנו ניסוי המשלב PCA במודל Decision Tree שלנו כדי להעריך אם הפחתת מימד יכולה לשפר את הביצועים שלו. על ידי שמירה על רמות שונות של (95%, 90%, 85% ו-80%), שאפנו לפשט את מערך הנתונים ולשפר את הדיוק. התוצאות הראו שאין שיפור ואפילו ירידה בדיוק, בעוד ש-PCA יכול לעזור בהפחתת הרעש ובשיפור הביצועים עבור דגמים פחות מורכבים, הפחתת יתר עלולה להוביל לאובדן מידע משמעותי.

```
Overall Results:
File: csvResults/features.csv
PCA with 95.0% variance retained
{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.41
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.51
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.66
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.58
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.64
PCA with 90.0% variance retained
{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.41
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.51
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.66
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.59
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.64
PCA with 85.0% variance retained
{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.41
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.51
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.65
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.58
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.64
PCA with 80.0% variance retained
{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.41
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.51
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.65
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.58
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.64
```

לסיכום שיפור ביצועים עם PCA - ניסויים אלו מדגיש את האיזון הדרוש בין הפחתת ממדים לבין שימור נתונים חיוניים לביצועי המודל.

ניסיון לשיפור תוצאות עם JL

ערכנו את הניסוי באמצעות הלמה של JL (Johnson-Lindenstrauss) להפחתת ממדיות כדי לראות אם צמצום מספר התכונות יכול לשפר את הביצועים. טכניקות הפחתת מימדיות כמו JL יכולות לסייע בהתמודדות עם נתונים בעלי ממדים גבוהים, ולשפר את היעילות וההכללה של המודל. עם זאת, קיימות סכנות הקשורות לגישה זו: הקטנת הממדים יותר מדי עלולה להוביל לאובדן מידע חשוב, וכתוצאה מכך לירידה בדיוק המודל. Johnson-Lindenstrauss lemma קובעת שניתן להטמיע קבוצה של נקודות במרחב בעל ממדים גבוהים לתוך מרחב ממדי נמוך יותר באופן שהמרחקים בין הנקודות כמעט נשמרים.

```
Overall Results:
File: /kaggle/input/features2/features.csv
JL with 153 components, k=3: 0.85
JL with 153 components, k=5: 0.84
JL with 153 components, k=7: 0.83
JL with 153 components, k=9: 0.83
JL with 153 components, k=13: 0.81
JL with 153 components, k=17: 0.80
JL with 145 components, k=3: 0.85
JL with 145 components, k=5: 0.84
JL with 145 components, k=7: 0.83
JL with 145 components, k=9: 0.82
JL with 145 components, k=13: 0.81
JL with 145 components, k=17: 0.80
JL with 137 components, k=3: 0.85
JL with 137 components, k=5: 0.84
JL with 137 components, k=7: 0.83
JL with 137 components, k=9: 0.82
JL with 137 components, k=13: 0.81
JL with 137 components, k=17: 0.80
JL with 129 components, k=3: 0.85
JL with 129 components, k=5: 0.84
JL with 129 components, k=7: 0.83
JL with 129 components, k=9: 0.83
JL with 129 components, k=13: 0.81
JL with 129 components, k=17: 0.80
```

ערכנו ניסוי המשלב JL במסווג ה-KNN שלנו כדי לקבוע אם הפחתת הממדים של מערך הנתונים שלנו יכולה לשפר את ביצועי המודל. התוצאות שלנו מראות שבעוד ש-JL שמר על דיוק בדרך כלל, ביצועי המודל ירדו מעט במספרים שונים של שכנים. לדוגמה, הדיוק של KNN עם 3 שכנים היה 86% ללא JL אך ירד ל-85% עם JL, ללא קשר למספר הרכיבים בשימוש. זה מצביע על כך שבעוד ש-JL יכול לעזור בהפחתת הממדיות, הוא עשוי גם להוביל להפחתה קלה ברמת הדיוק, ולהדגיש את הפשרה בין יעילות חישובית וביצועי מודל.

ערכנו ניסוי המשלב JL במודל Decision Tree שלנו כדי לבדוק האם אפשר להגיע לתוצאות טובות יותר. התוצאות שלנו הראו שבעוד ש-JL שמרה על רמת דיוק דומה למודל המקורי, היא לא השיגה אותה באופן עקבי, מה שמעיד על כך שייתכן שחלק מהמידע החיוני אבד בתהליך. לדוגמה, ללא JL, המודל עם הפרמטרים `{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}` השיג דיוק של 0.66, בעוד עם JL, הדיוק נע בין 0.61 ל-0.62.

```
Overall Results:
File: csvResults/features.csv
JL with 153 components, {'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.39
JL with 153 components, {'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.48
JL with 153 components, {'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.62
JL with 153 components, {'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.54
JL with 153 components, {'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.61
JL with 145 components, {'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.38
JL with 145 components, {'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.47
JL with 145 components, {'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.61
JL with 145 components, {'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.52
JL with 145 components, {'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.59
JL with 137 components, {'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.36
JL with 137 components, {'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.46
JL with 137 components, {'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.61
JL with 137 components, {'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.52
JL with 137 components, {'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.59
JL with 129 components, {'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.37
JL with 129 components, {'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2}: 0.48
JL with 129 components, {'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1}: 0.61
JL with 129 components, {'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf': 4}: 0.53
JL with 129 components, {'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf': 2}: 0.58
```

הוספת פיצ'רים:

- **מרכז ספקטרלי:** מייצג את "מרכז המסה" של הספקטרום, ומצביע על המקום שבו ממוקם מרכז האנרגיה. הוא יכול לספק תובנות לגבי הבהירות של הצליל.
- **רוחב ספקטרלי:** מודד את רוחב הספקטרום. הוא עוזר לקבוע את הטקסטורה של הצליל, כאשר רוחבים ספקטרליים רחבים יותר מעידים על צלילים מורכבים יותר.
- **שטיחות ספקטרלית:** מודד עד כמה הספקטרום שטוח (או דמוי רעש). ערכים גבוהים יותר מעידים על אותות דמויי רעש, בעוד שערכים נמוכים יותר מעידים על אותות טונאליים.

ערכנו ניסוי על מודל NN עם עוד מאפיינים לדאטה, ניתן לראות בתוצאות אחוז הדיוק עלה לא בצורה משמעותית.

```
Epoch [10/50], Loss: 0.6433
Epoch [20/50], Loss: 0.5425
Epoch [30/50], Loss: 0.3754
Epoch [40/50], Loss: 0.0942
Epoch [50/50], Loss: 0.3138
Test Accuracy: 85.43%
```

ערכנו ניסוי על מודל KNN עם עוד מאפיינים לדאטה, ניתן לראות בתוצאות אחוז הדיוק לא השתנה.

```
Overall Results:
File: csvResults/more_features.csv
k=3: 0.86
k=5: 0.85
k=7: 0.84
k=9: 0.84
k=13: 0.82
k=17: 0.81
```

ניתוח תוצאות

K-Nearest Neighbors:

בבדיקה על ערכים שונים של שכנים שביצענו על מודל KNN, קיבלנו את הדיוק הבאה:
כאשר מספר השכנים שהאלגוריתם בדק מסביב לנקודה היו:
שכנים: 3 ,הדיוק היה: 86%
שכנים: 5 ,הדיוק היה: 85%
שכנים: 7 ,הדיוק היה: 84%
שכנים:9 ,הדיוק היה: 83%
שכנים: 13 ,הדיוק היה: 82%
שכנים: 17 ,הדיוק היה: 81%

הדיוק בטוב ביותר הושג עבור 3 שכנים. כאשר מספר השכנים קטן, המודל נוטה להיות יותר רגיש לשינויים בנתונים. זה יכול להוביל לדיוק גבוה יותר אם הנתונים הם יחסית שונים. בנוסף KNN רגיש למרחקים בין הדוגמאות. אם יש בתכונות, שונות רבה בנתונים, דיוק המודל יכול לרדת. השימוש ב-StandardScaler מסייע בהתמודדות עם בעיה זו, אך ייתכן שעדיין יש קושי.

Decision Tree:

{'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf: 1}
=> Accuracy = 0.35
{'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf: 2 }
=> Accuracy = 0.44
{'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf: 1}
=> Accuracy = 0.66
{'max_depth': 7, 'criterion': 'gini', 'min_samples_split': 10, 'min_samples_leaf: 4}
=> Accuracy = 0.53
{'max_depth': 10, 'criterion': 'entropy', 'min_samples_split': 10, 'min_samples_leaf: 2}
=> Accuracy = 0.64

max_depth: קובע את עומק העץ.
criterio: הקריטריון לקביעת השאלות בעץ.
min_samples_split: מספר הדוגמאות המינימלי הנדרש כדי לחלק נוד.
min_samples_leaf: מספר הדוגמאות המינימלי בנוד העליון.

התוצאות מראות כי קונפיגורציות שונות ב max_depth, criterion, min_samples_split, ו-min_samples_leaf, מובילות לרמת דיוק שונה. דיוקים נמוכים (0.35, 0.44) עשויים לנבוע מפרמטרים מגבילים מדי, כמו עומק נמוך או חיתוך מינימלי גבוה, שמגבילים את יכולת המודל ללמוד מהנתונים. דיוקים גבוהים יותר (0.66, 0.64) קשורים ככל הנראה למודלים מורכבים יותר (עומק גבוה יותר) שיכולים לתפוס תבניות בצורה יותר יעילה.
ניתן לראות בהמשך כי Random Forest נותן תוצאות טובות יחסית מ-Decision Tree ולכן ההנחה העיקרית שלנו שמדובר בהתאמת יתר ש-Random Forest מטפל בזה באמצעות ריבוי עצים

Random Forest:

בבדיקה על ערכים שונים של פרמטרים של העץ שביצענו על מודל Random Forest, קיבלנו את הדיוק הבאים:

{'n_estimators': 100, 'max_depth': 3, 'criterion': 'gini', 'min_samples_split': 2,
min_samples_leaf': 1} => Accuracy = 0.52,

{'n_estimators': 150, 'max_depth': 5, 'criterion': 'gini', 'min_samples_split': 4, 'min_samples_leaf': 2} => Accuracy = 0.64,
{'n_estimators': 100, 'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1} => Accuracy = 0.80,
{'n_estimators': 150, 'max_depth': None, 'criterion': 'entropy', 'min_samples_split': 4, 'min_samples_leaf': 2} => Accuracy = 0.80,
{'n_estimators': 100, 'max_depth': 3, 'criterion': 'entropy', 'min_samples_split': 2, 'min_samples_leaf': 1} => Accuracy = 0.53
{'n_estimators': 150, 'max_depth': 7, 'criterion': 'entropy', 'min_samples_split': 4, 'min_samples_leaf': 2} => Accuracy = 0.72

n_estimators: מספר העצים ביער,
max_depth: העומק המרבי של כל עץ החלטות,
criterion: הפונקציה המשמשת למדידת איכות הפיצול בכל עץ החלטות.
min_samples_split: המספר המינימלי של דגימות הנדרש לפיצול צומת פנימי,
min_samples_leaf: המספר המינימלי של דגימות הנדרש להיות בצומת עלה.

דיוק המודל Random Forest נע בין 0.52 ל- 0.80, כאשר הדיוק הגבוה ביותר הושג באמצעות קונפיגורציות עם max_depth=None ו- criterion='entropy'.
יותר עצים בדרך כלל משפרים את הביצועים אך גם מגבירים את זמן החישוב. הגבלה על עומק העץ (למשל, max_depth=3) עלולה להוביל לתת-התאמה, בעוד ש- max_depth=None מאפשר למודלים להיות מורכבים יותר, מה שעשוי לשפר את הדיוק. שימוש ב- entropy הניב דיוק גבוה יותר מאשר gini. ערכים גבוהים יותר עבור min_samples_split ו- min_samples_leaf יכולים למנוע התאמה יתרה, אך עלולים להגביל את יכולת המודל ללכוד תבניות.

Support Vector Machine:

בבדיקה על סוגי kernel שונים קיבלנו את הדיוקים האלו:

kernel=linear => Accuracy = 0.71
kernel=rbf => Accuracy = 0.76
kernel=poly => Accuracy = 0.45
kernel=sigmoid => Accuracy = 0.56

linear: פונקציית ליבה זו משמשת כאשר הנתונים ניתנים להפרדה ליניארית. זה יוצר גבול החלטה ליניארי.
rbf: ידוע גם בשם הגרעין גאוסי, הוא ממפה נתונים לתוך מרחב ממדי גבוה יותר שבו קל יותר למצוא הפרדה ליניארית. זה יעיל ללכידת קשרים לא ליניאריים.
poly: ליבה זו מייצגת נתונים באמצעות פונקציות פולינומיות בדרגה מסוימת. זה מאפשר מודלים של קשרים לא ליניאריים עם מורכבות פולינומית.
sigmoid: ליבה זו דומה לפונקציית ההפעלה המשמשת ברשתות עצביות. הוא ממפה את תכונות הקלט לפונקציית משיק היפרבולית, ומציגה אי-לינאריות.

הקרנל הליניארי מתאים להפרדת נתונים בקו ישר, והתוצאה של 0.71 מצביעה על כך שהוא מזהה דפוסים, אך לא בצורה אופטימלית, מה שמעיד על כך שהנתונים לא ניתנים להפרדה לינארית בקלות. קרנל RBF מיועד לנתונים שאינם ליניאריים, והוכיח את עצמו עם תוצאה של 0.76, מה שמעיד על יכולתו ללכוד דפוסים מורכבים. הקרנל הפולינומיאלי מתאים לתלות פולינומית בין תכונות, התוצאה הנמוכה של 0.45 מצביעה על כך שהקרנל לא מצליח ללמוד מהדפוסים. הקרנל הסיגמואידי פחות נפוץ, והצגת תוצאה של 0.56 מצביעה על

חוסר יכולת ללכוד דפוסים בצורה יעילה. קרנל RBF הוא הבחירה המומלצת למקרה שלנו, בהתבסס על התוצאות והמאפיינים של קרנלים.

Neural Network:

דיוק שקיבלנו - 84%

גם training וגם validation משתפרים בתחילה, לאחר מספר מסוים של איפוקים, דיוק training ממשיך לעלות בעוד שדיוק validation פוחת. זה מצביע על כך שהמודל עשוי להתאים יתר על המידה לנתוני האימון. התאמת יתר מתרחשת כאשר המודל לומד את נתוני האימון טוב מדי. אולי הוספת רעשים לאודיו היה יכול לפתור את הבעיה. רשתות נוירונים מסוגלות להכליל טוב יותר על פני נתונים חדשים בזכות היכולת שלהן ללמוד תכנים בצורה עמוקה. מודלים כמו KNN תלויים מאוד במרחקים בין דוגמאות ולפעמים מתקשים להתמודד עם נתונים חדשים (כמו שינוי קולות בהקלטה דיבור חלש יותר או מבטא) אם הם לא דומים לדוגמאות שהיו בתהליך הלמידה.