איילת מויאל 204687685 , ליאל בנימין 319081600 , נטליה קטייב 323225409 , יונתן גרינשפן 204377899

סדנת הכנה לפרוייקט

עבודה 4 – ניתוח נתונים וחיזוי

תוכן עניינים

הקדמה:

בשנים האחרונות, תחום החיזוי התפתח בצורה דרסטית עקב התקדמות טכנולוגית בתחום למידת המכונה וגידול הערך העסקי של חברות רבות.
שלב בעל תרומה עיקרית לפתרון בעיות החיזוי הינו עיבוד מקדים עבור המידע המקורי, ומהווה גורם מכריע לאיכות תוצאות המודל.
בחרנו להתמודד עם ארבעה סוגים של בעיות סיווג עבור ה- dataset המקורי.
עבור ארבעת האלגוריתמים הללו, ביצענו תהליך שכולל מספר שלבים, החל מחלוקת ה- dataset ל-90% training ו-10% testing, 10 fold Cross Validation.

על מנת לעשות את בחירת הhyper parameters הטובים ביותר עבור כל אלגוריתם, האלגוריתם ביצע עבור כל מסווג מספר הרצות שונות עם מגוון hyper parameters שונים, השווה בין התוצאות השונות כדי לדעת אילו פרמטרים יתנו לנו את הפתרון האופטימלי ולבסוף השתמשנו בפרמטרים שהאלגוריתם החזיר.

בהסתמך על מטלת הסקירה הספרותית בחרנו לבצע את הניסויים השונים בעזרת האלגוריתמים KNN, ANN, Random Forest, ו-Decision Tree , בעבודות המחקר שסקרנו. עבור כל אלגוריתם, התאמנו את המימוש התואם לו מספריית sklearn .

תיאור הנתונים:

תיאור הנתונים ותהליך ה- Preprocessing
מאגר הנתונים הכיל 150 טבלאות, אשר מתוכן השתמשנו ב10 הטבלאות הנ"ל:

שם הdata base	גודל הטבלה	מספר התכונות
ar4	108	29
bank	4522	16
blood	749	4
breast_cancer	287	9
chess	3197	36
haserman survival	307	3
heart Hungarian	295	12
abalone	4178	8
parkinson	196	22
vertebral_column_2classes	311	6

בחרנו בdataset המכיל סיווגים בינאריים או לא בינאריים, ניסינו ליצור אוסף נתונים שיכיל כמה שיותר מידע ומגוון נתונים, כדי לבחון איזה אלגוריתם נותן את הסיווג הטוב ביותר לכל מידע אשר יהיה בdataset.
כמו כן, ווידאנו כי כמות התכונות וגודל ה dataset יהיה שונה בין כל dataset ובכך לגרום לשינוי בין אופן הרצת המודלים.

תיאור האלגוריתמים/ מודלים:

- **Random Forest :** אלגוריתם נפוץ בפתרון בעיות שונות בלמידת מכונה כגון סיווג, אשר מורכב ממספר עצי החלטה שנבנים בזמן ה- Training המשולבים למודל אחד ששואף להוות שיפור עבור ביצועי כל עץ בנפרד.

הערכים שנבדקו	הסבר
max_features	מספר התכונות שיש לקחת בחשבון כאשר מחפשים את הפיצול הטוב ביותר.
class_weight	balanced_subsample/balanced - האם המשקלים מחושבים על סמך הbootstrap עבור כל עץ שגדל.

- **KNN:** אלגוריתם סיווג או רגרסיה פשוט יחסית בתחום למידת מכונה, אשר מתבסס על התצפיות הקרובות ביותר במרחב הפיצ'רים.
זהו אלגוריתם לימוד מבוסס מופעים אשר לרוב עושה שימוש במטריקת המרחק האוקלידי.
כמו כן, הדיוק של הסיווג עלול להיפגע מרגישות האלגוריתם למבנה המקומי של הנתונים או מנוכחותם של רעש ותכונות לא רלוונטיות.

הערכים שנבדקו	הסבר
Algorithm	אלגוריתם המשמש לחישוב השכנים הקרובים ביותר(auto/brute /kd_tree /ball_tree).
N_neighbors	מספר השכנים.
Weights	משקלים אחידים/ משקל גבוה לשכנים הרחוקים יותר ולהפך.
Leaf_size	גודל העלים שהועבר לאלגוריתם הנבחר. זה יכול להשפיע על מהירות הבנייה והשאילתה.

- **ANN:** מודל מתמטי שפותח בהשראת תהליכים קוגניטיביים ומוחיים המשמש בעיקר בתחום למידת מכונה.
מודל זה מורכב מיחידות רבות של מידע המקושרות ביניהן ומכיל שלושה סוגים של שכבות Input layer, Hidden Layer, Output layer.

הערכים שנבדקו	הסבר
Activation	פונקציית הפעלה עבור השכבה הנסתרת(relu / tan/ logistic /identity).
Learning_rate	קצב למידה – constant (קבוע) / invscaling (מקטין בהדרגה) / adaptive (כל עוד יש ירידה הוא קבוע אחרת מחולק ל-5).

- **Decision Tree:** אלגוריתם אשר משמש לסיווג או חיזוי ערכו של משתנה מטרה ומספק מודל
שנבנה בעזרת סט האימון, כאשר הפיצ'רים (צמתי הפיצול) בעץ מסודרים לפי סדר החשיבות של הרווח מהמידע הצפוי מהם .

הערכים שנבדקו	הסבר
Criterion	הפונקציה למדידת איכות הפיצול(gini/ entropy /log_loss).
Splitter	האסטרטגיה המשמשת לבחירת הפיצול בכל צומת(random /best).
Max_features	מספר התכונות שיש לקחת בחשבון כאשר מחפשים את הפיצול הטוב ביותר.
Class_weight	balanced_subsample/balanced - האם המשקלים מחושבים על סמך הbootstrap עבור כל עץ שגדל.

תיאור הרצת ניסוי/ הערכה:

- תחילה בחרנו קובץ csv מתוך ה-database לדוגמא –abalone.csv והעלנו אותו לcolab.
- יצרנו dataframe ריק שבו נשמור את התוצאות.
- ביצענו ניקוי נתונים - בדיקת ערכי null, הורדת רשומות זהות, מחיקת הרשומה הראשונה כי אין צורך בכותרות התכונות.
- ערבבנו את הרשומות כדי למנוע מצב שבו המודל מותאם יתר על המידה לאוסף מסוים של נתונים.
- חילקנו את הנתונים ל X- התכונות ו Y- מטרות (מה שאנו מנסים לחזות).
- נעזרנו בפונקציות עזר כגון:

- scaling_data() שמתאים את הנתונים שיהיו באותו טווח.
- Train_alghorithm() מקבלת את נתונים והמודל ומבצעת fit (למידה).
- calculate_pecision_recall() מחשבת את המדדים fpr ו- tpr שבעזרתם נוכל ליצור auc_roc_curve ונחשב את ה-auc_score .
- binary_transformer() נקראת כאשר יש Database עם סיווג שאינו בינארי, היא עושה טרנספורמציה על הסיווגים והופכת אותם לבינארי עבור חישוב roc_curve.
- plot_roc_curve() מציירת את הגרף roc_curve עבור כל הרצה.
- calculate_hyperparameters() מקבלת מודל ואת הפרמטרים ומחזירה את הפרמטרים שמביאים לערכים הטובים ביותר, את המודל נעדכן כך שהפרמטרים שלו יהיו הפרמטרים האלו.

דוגמא לחישוב הערכים הטובים ביותר עבור KNN:

```
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 15, 'leaf_size': 40, 'algorithm': 'kd_tree'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'uniform', 'n_neighbors': 15, 'leaf_size': 20, 'algorithm': 'brute'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 15, 'leaf_size': 40, 'algorithm': 'kd_tree'}
Best Hyperparameters for KNN are:  {'weights': 'distance', 'n_neighbors': 30, 'leaf_size': 20, 'algorithm': 'auto'}
```

- שמרנו במילון את כל המודלים שאנחנו משווים ביניהם ואת הפרמטרים השונים שאיתם נבצע קומבינציות שונות על מנת למצוא את הערכים האופטימליים.
- Calculate_ML_measurements() הפונקציה הראשית אשר קוראת לכל פונקציות העזר, מחשבת את התוצאות ושומרת אותם בdataframe.
  - תחילה בחרנו K=10 כלומר, 10 fold Cross Validation – הרצנו כל 10 dataset פעמים כאשר בכל הרצה יש 90% train ו-10% test ובכל הרצתה החלוקה תהיה בעלת נתונים שונים.
  - הפונקציה מחשבת את המדדים הבאים: Accuracy, TPR, FPR, Precision, AUC ROC Curve, AUC Precision-Recall, Training Time, Inference time for 1000 instances על מנת להשוות בין האלגוריתמים.

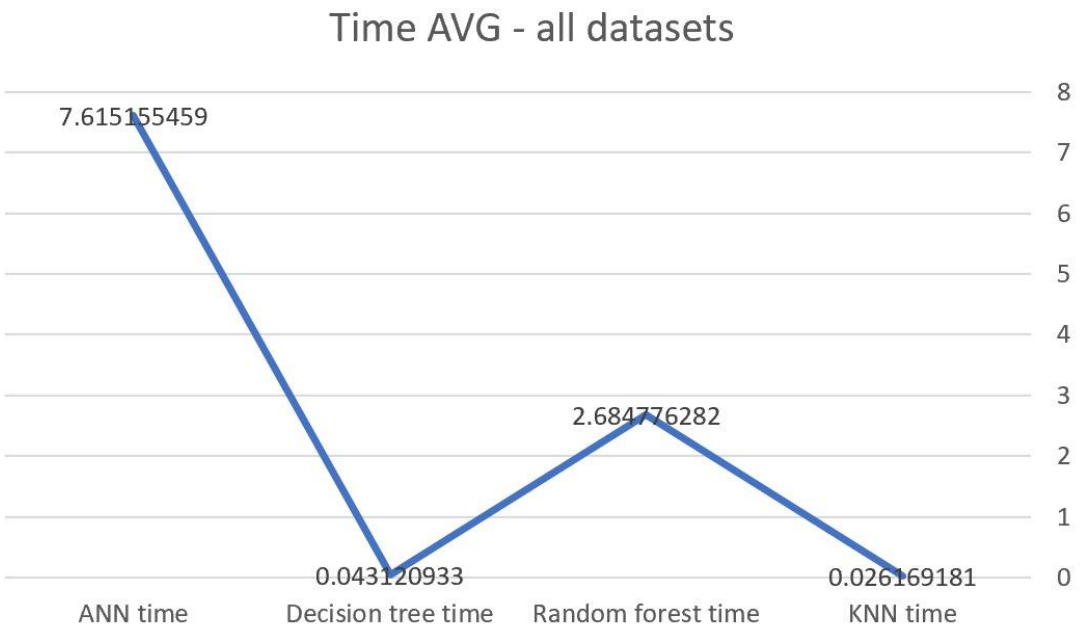
- לבסוף העברנו את ה-dataframe לקובץ csv המכיל את כל התוצאות.
דוגמא לנתונים בקובץ הCSV של chess-krvkp:

	measurment	KNN	Random forest	Decision tree	ANN
0	Avg Acuracy	0.949003	0.990928	0.960267	0.984671
1	Acuracy STD	0.010297	0.003436	0.0069	0.011916
2	Avg precision	0.950724	0.994078	0.955839	0.982014
3	precision STD	0.018947	0.005766	0.01073	0.020157
4	Avg recall	0.942425	0.986822	0.961539	0.98612
5	recall STD	0.012802	0.005502	0.011911	0.01018
6	Avg f1 score	0.948991	0.990926	0.960272	0.984674
7	f1 score STD	0.010295	0.003436	0.006906	0.011906
8	Avg auc roc curve score	0.948747	0.990718	0.960363	0.984691
9	roc score STD	0.010244	0.003457	0.007009	0.011596
10	TPR	0.942425	0.986822	0.961539	0.98612
11	FPR	0.04493	0.005386	0.040812	0.016738
12	avg inference time	0.011537	0.004255	0.000047	0.000403
13	inference STD	0.006287	0.000218	0.000005	0.000231
14	train time	0.088894	2.426772	0.038382	36.234687

- את הקוד הרצנו עבור כל 10 הdataset שבחרנו.

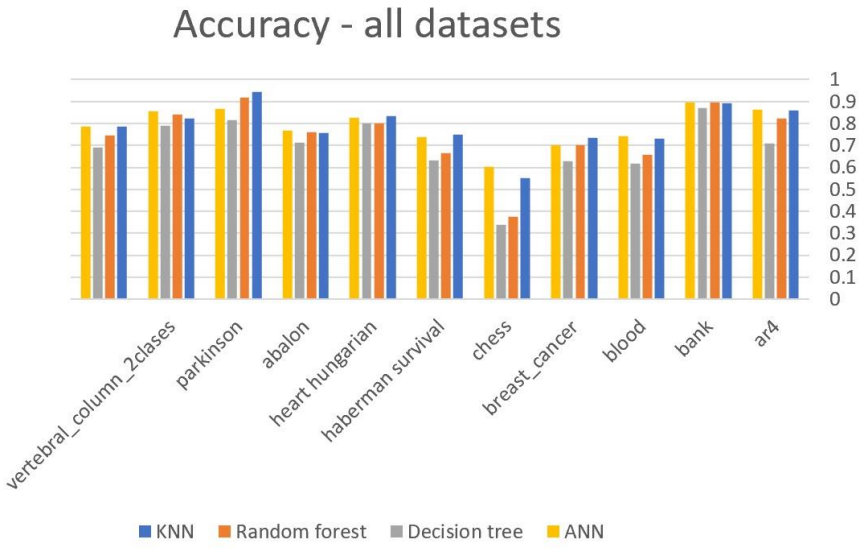
תוצאות ומסקנות:

תוצאות הזמן הממוצע של כל מודל על כל הdataset שבחרנו:



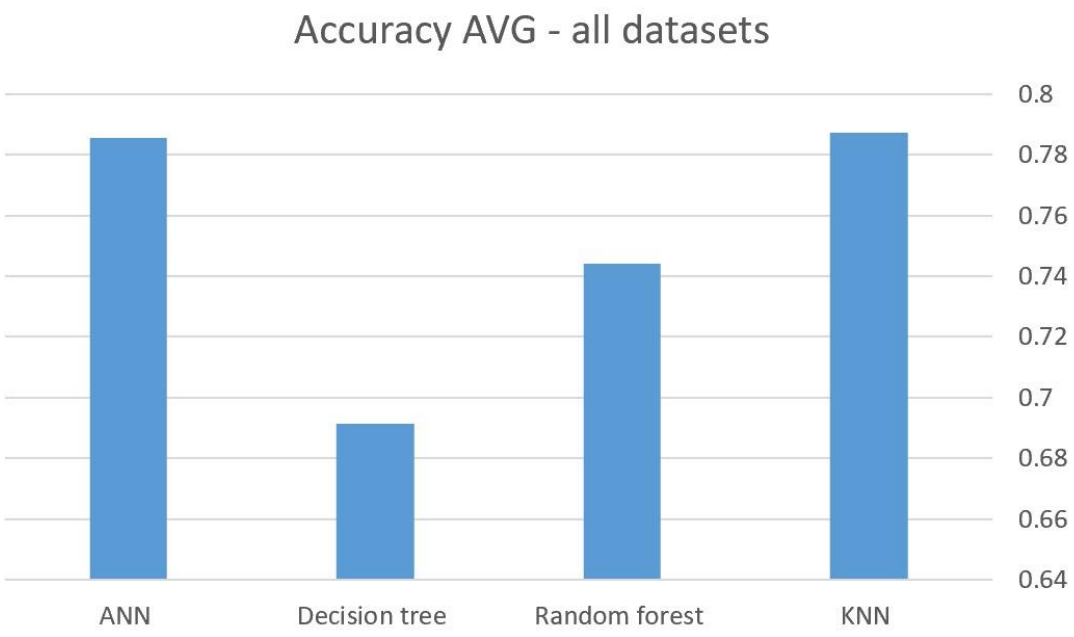
ניתן לראות שהזמן הממוצע של KNN הוא הנמוך ביותר, זה נובע מכך שהוא אלגוריתם פשוט, ושל ANN הוא הגבוה ביותר מכיוון שיש צורך בטעינה ושמירה של נתונים וזה גוזל זמן רב.

תוצאות הaccuracy כל הdataset שבחרנו:



ניתן לראות ש הaccuracy של chess הכי נמוך מכיוון שהוא עם מספר התכונות הגבוה ביותר (36).

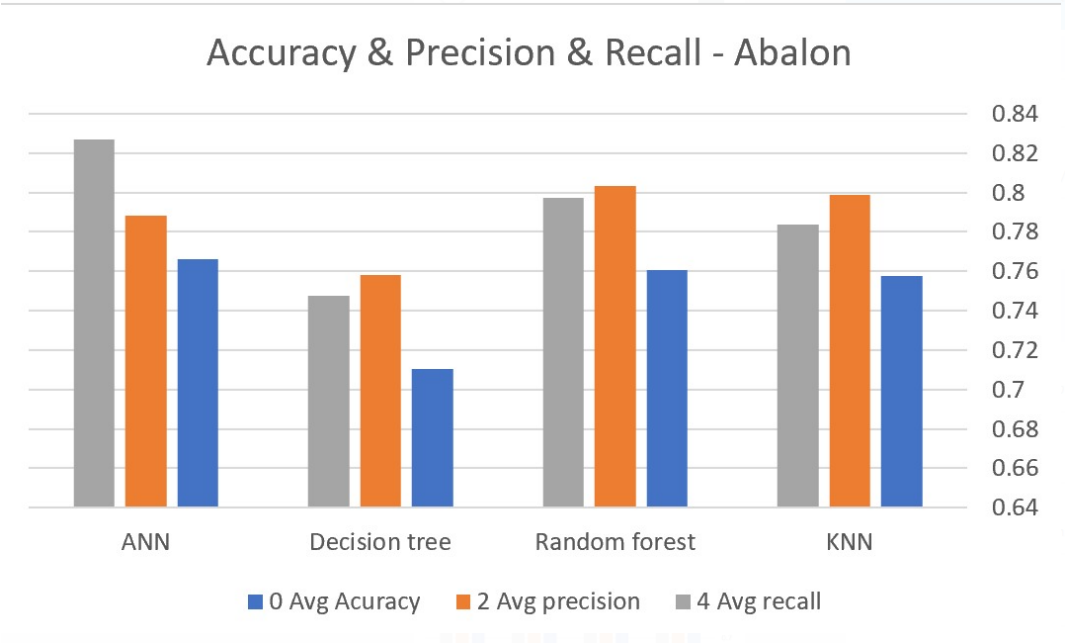
תוצאות ממוצע הaccuracy עבור כל מודל:



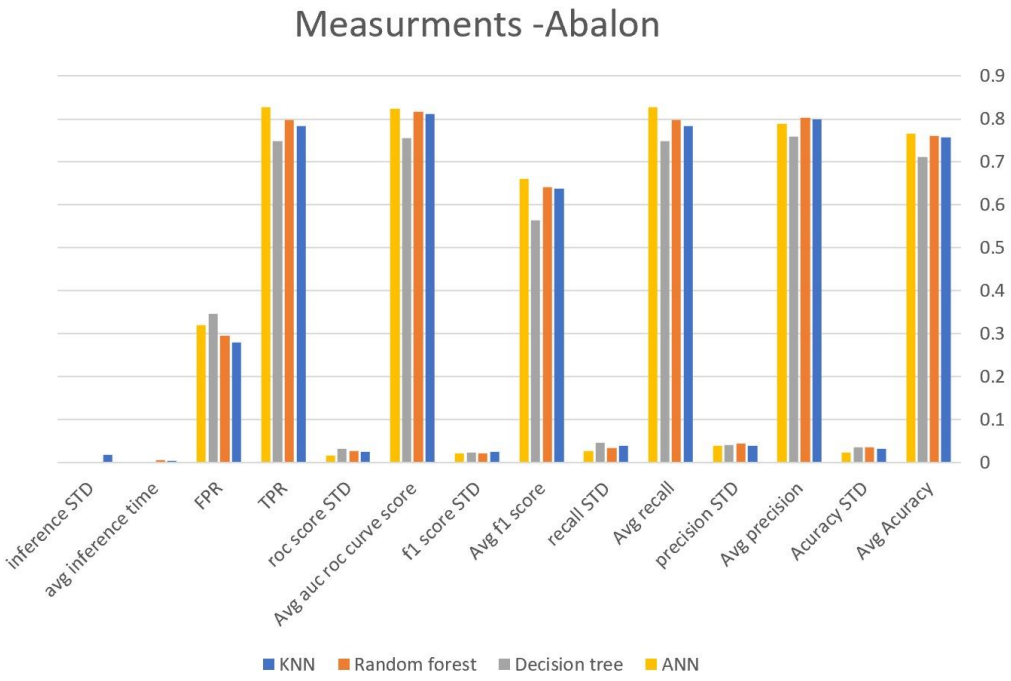
ניתן לראות ש הaccuracy של decision tree הוא הכי נמוך, כנראה מכיוון שיש לו נטייה לבצע overfitting ביחס לשאר המודלים.

ניקח לדוגמא את הקובץ של abalon ונדגים עליו תוצאות נוספות:

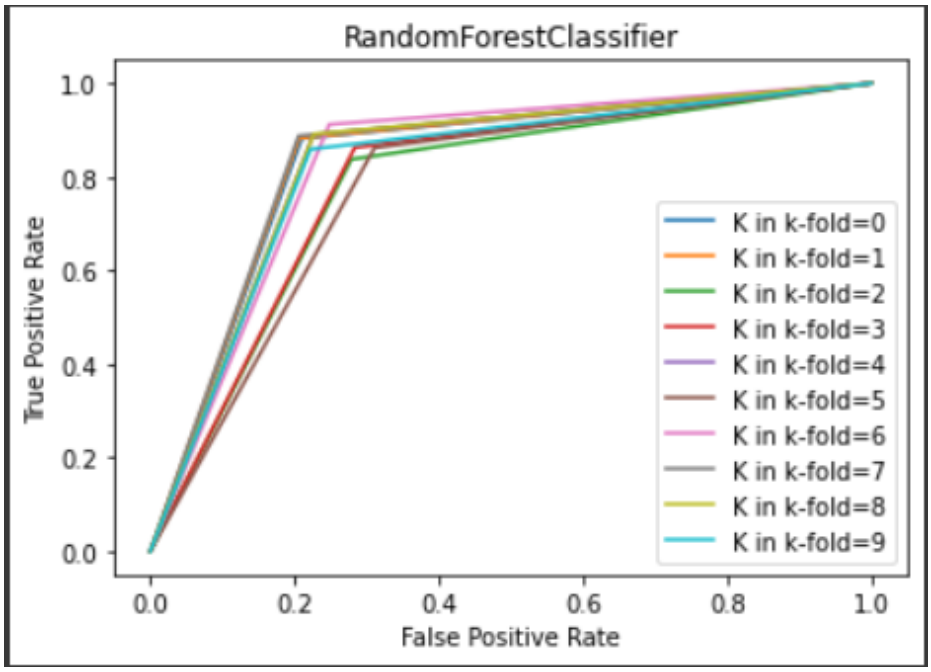
תוצאות המדדים accuracy, precision, recall של abalon:

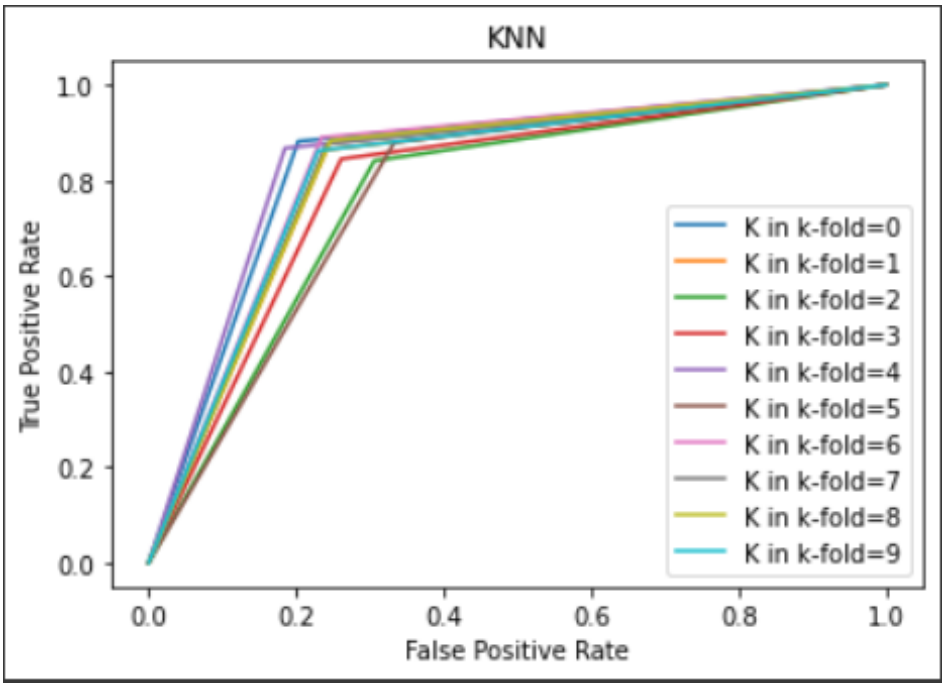


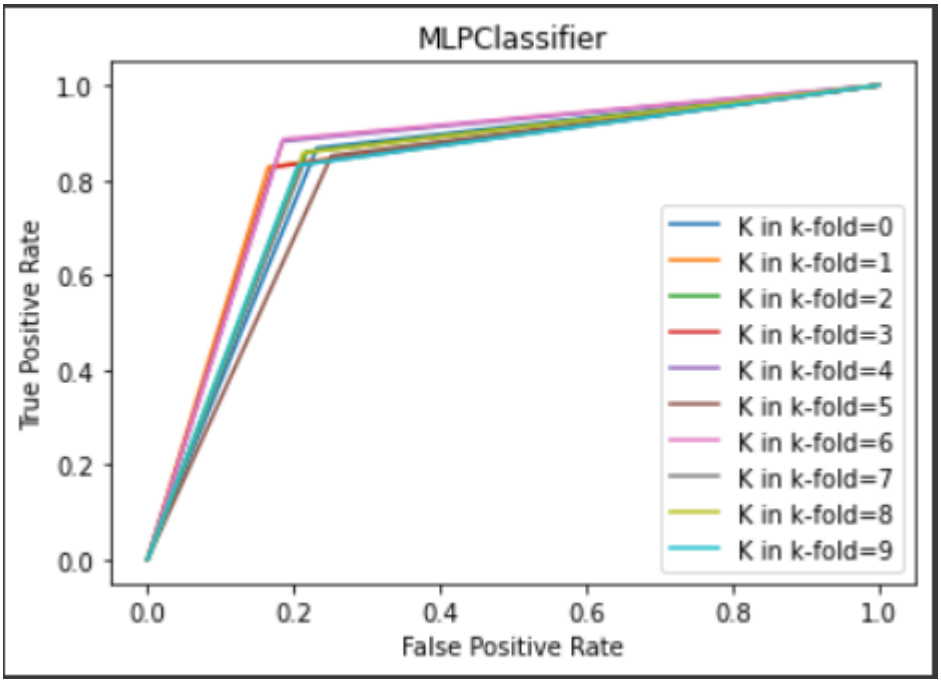
תוצאות לmeasurments של abalon:

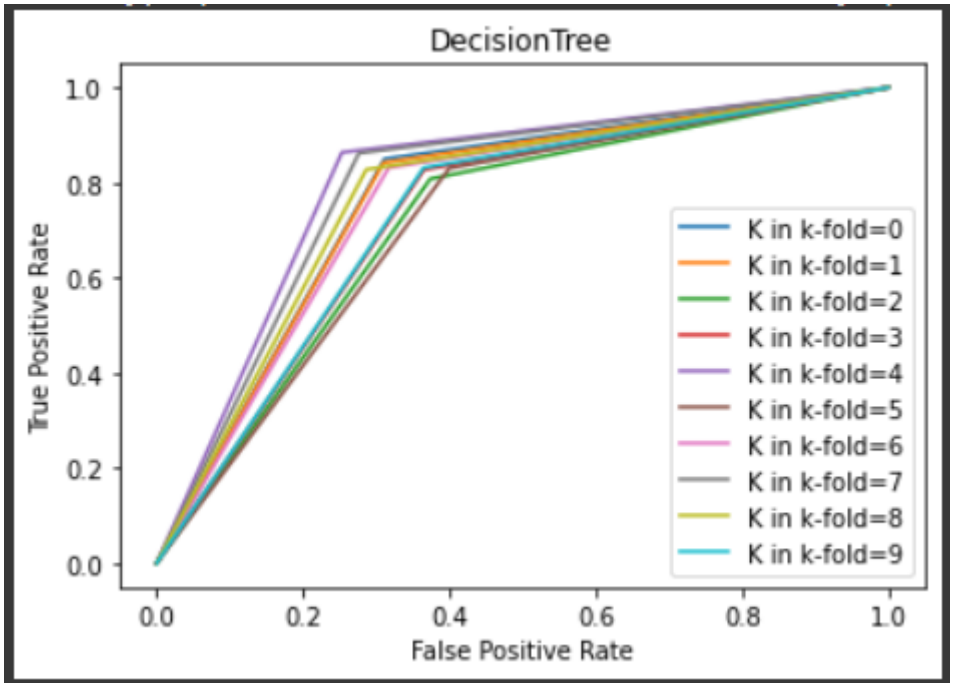


תוצאות הauc roc curve של abalon עבור כל מודל:






MLPClassifier
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
K in k-fold=0
K in k-fold=1
K in k-fold=2
K in k-fold=3
K in k-fold=4
K in k-fold=5
K in k-fold=6
K in k-fold=7
K in k-fold=8
K in k-fold=9


DecisionTree
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
K in k-fold=0
K in k-fold=1
K in k-fold=2
K in k-fold=3
K in k-fold=4
K in k-fold=5
K in k-fold=6
K in k-fold=7
K in k-fold=8
K in k-fold=9

מתוך התוצאות שהתקבלו ניתן לראות כי ביצועי המודל KNN היו הטובים ביותר, נראה כי מבחינת הדיוק הביצועים שלו היו הטובים ביותר, ומבחינת זמן הוא היה הכי נמוך, כלומר הוא המודל המהיר ביותר.
ביצועי KNN טובים יותר כאשר הdataset קטן יותר ולכן כנראה אם היינו בוחרים dataset מאוד גדול אז התוצאות היו יכולות להשתנות.

ביצועי המודל של Decision tree הם הגרועים ביותר על סמך המדדים שחישבנו, מבחינת הדיוק הביצועים שלו היו הכי נמוכים, נראה כי מודל זה הכי פחות יעיל.
אם היינו רוצים לשפר את הדיוק של הDecision tree היינו יכולים להשתמש באלגוריתם boosting.

התוצאות תלויות בבחירה האקראית של הtraining וה-testing, תלוי בערבוב הרנדומאלי שעשינו לdataset, ותלוי בפרמטרים שהפונקציה hyperparameters optimization תחזיר לנו ולכן בכל ריצה התוצאות יכולות להיות שונות.

בונוס:

**** הערה חשובה לבודק: בכדי להריץ את הקובץ הבונוס נדרש להשתמש בקובץ "all-results.csv". ****

לאחר מחקר שערכנו בנוגע ל-friedman test ו-post hoc analysis וביצענו בעזרתם מבחני מובהקות סטטיסטיים בכדי לבחור את המודל הטוב ביותר. השתמשנו במבחן זה בכדי להבין האם קיים מודל כלשהו מבין המודלים שבחרנו שיש לו ערכים השונים בצורה מובהקת מהמודלים האחרים שבנינו. בחרנו לבצע את המבחן שלנו עבור שני מדדים – accuracy ו-time, ועבורם בנינו קובץ csv המכיל את מדד ה-accuracy, ה-time עבור כל אחד מהמודלים . בנוסף בחרנו את רמת הסמך להיות 0.05. לאחר מכן הפכנו נתונים אלו למערכים, והרצנו על כל אחד מהם את מבחן friedman. אחרי המבחן, קיבלנו את ערכי p-value הבאים:

```
Friedman Test Accuracy:
FriedmanchisquareResult(statistic=19.560000000000002, pvalue=0.00020937828737529863)
```

```
Friedman Test Time:
FriedmanchisquareResult(statistic=28.079999999999984, pvalue=3.494281590861251e-06)
```

ראינו כי גם עבור בחירת ערך קטן יותר של ה- p-value התוצאות עדיין נותרות מובהקות ויש הבדל בין המודלים השונים. לאחר מכן, כאשר ידוע לנו מה ההבדלים בין המדדים השונים של המודלים, השתמשנו ב- post hoc analysis על מנת שנוכל להבין עבור כל מודל את ההתנהגות הפנימית שלו. גם עבור post hoc analysis נבחר את רמת הסמך להישאר 0.05.

```
Friedman Test Accuracy:
FriedmanchisquareResult(statistic=19.560000000000002, pvalue=0.00020937828737529863)
Post hoc Accuracy:
                    KNN  Random forest  Decision tree       ANN
KNN            1.000000       0.900000       0.516551  0.900000
Random forest  0.900000       1.000000       0.670273  0.900000
Decision tree  0.516551       0.670273       1.000000  0.823993
ANN            0.900000       0.900000       0.823993  1.000000
```

```
Friedman Test Time:
FriedmanchisquareResult(statistic=28.079999999999984, pvalue=3.494281590861251e-06)
Post hoc Time:
               KNN  Random forest  Decision tree  ANN
KNN            1.0            0.9            0.9  0.9
Random forest  0.9            1.0            0.9  0.9
Decision tree  0.9            0.9            1.0  0.9
ANN            0.9            0.9            0.9  1.0
```

אנו יודעים שברמת מובהקות של לפחות 0.05 ישנו הבדל בין ארבעת המודלים שבחרנו, ולכן נוכל להשוות ביניהם בעזרת הממוצע שלהם ולפי התוצאות שקיבלנו בבדיקות.

KNN average accuracy	0.78711175
Random forest average accuracy	0.74399676
Decision tree average accuracy	0.69153426
ANN average accuracy	0.78548149
KNN average time	0.02616918
Random forest average time	2.68477628
Decision tree average time	0.04312093
ANN average time	7.61515546