ML Test

August 2023

Objectives

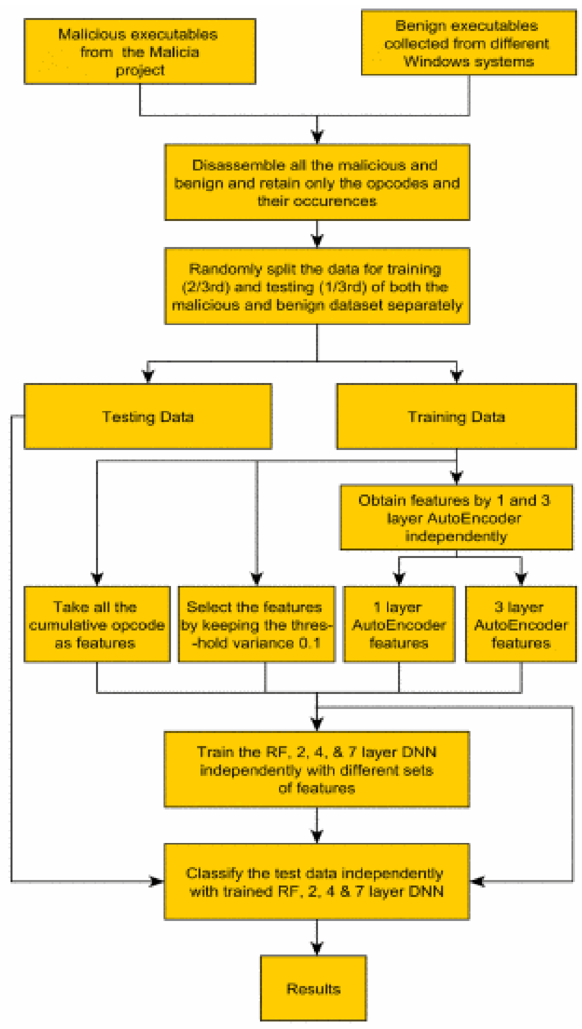
1. To determine the **strengths, weaknesses, and feasibility of the candidate datasets** in building an ML model.
2. To determine and test primary techniques in **dataset preparation by means of data pre-processing and cleaning.**
3. To determine the **basic processes and factors that may affect building an ML model**.
4. To determine **which of the candidate datasets perform best regardless of ML model**.
5. To determine **which of the selected models perform best on each dataset**.
6. To determine **if ensemble/boosted ML models are more beneficial than traditional ML models.**

Framework

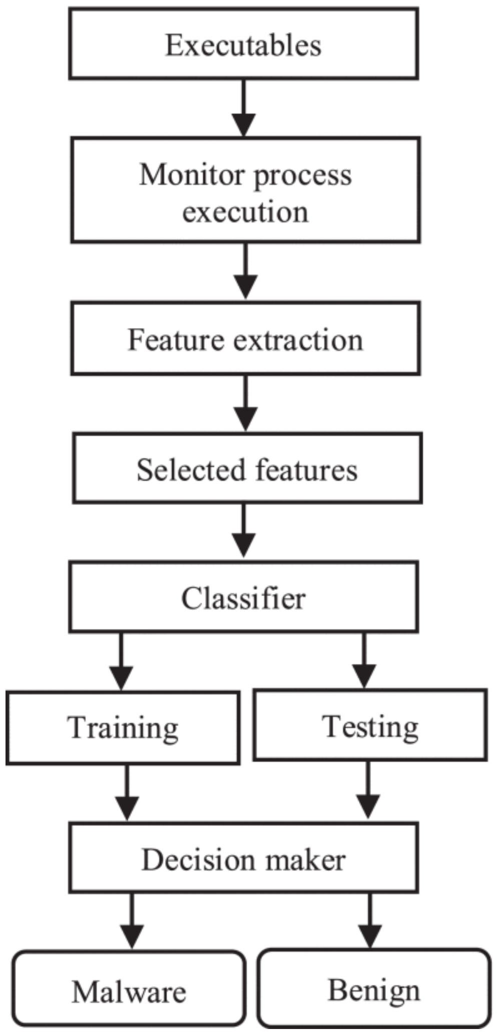
Reference Frameworks

[1] M. Sewak, S. K. Sahay, and H. Rathore, “Comparison of Deep Learning and the Classical Machine Learning Algorithm for the Malware Detection,” in 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan: IEEE, Jun. 2018, pp. 293–296. doi: 10.1109/SNPD.2018.8441123.

[2] O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” IEEE Access, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.



[1]

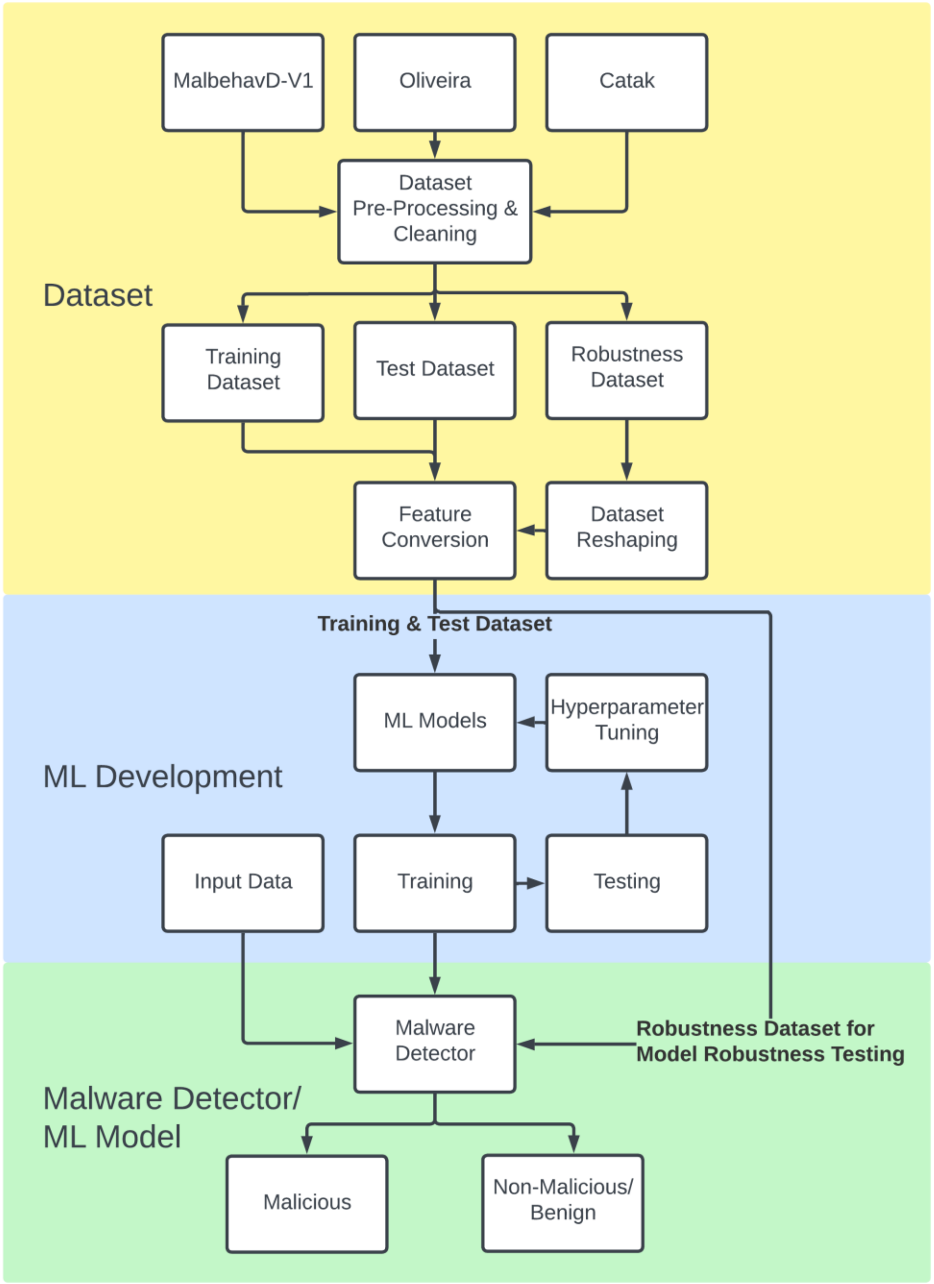


[2]

Proposed Framework

The proposed framework consists of three major segments which are the dataset, ML training, and ML detector.

1. **Dataset**
    1. Dataset Examination
    2. Dataset Preparation (Data Preprocessing & Cleaning)
    3. Dataset Division/Allocation (Training & Test; Robustness Dataset)
    4. Reshaping (for Robustness Dataset)
    5. Feature Conversion (String to int)
2. **ML Training** – Training, Testing, Tuning
3. **ML Detector**
    1. Trained ML Model (Malware Detector)
    2. Malicious and Benign/Non-Malicious Classifier



Tools

1. Python 3
2. Anaconda
3. Jupyter Notebook
4. Python 3 Libraries: Scikit-Learn, Pandas, Numpy

Machine Learning with Scikit-Learn

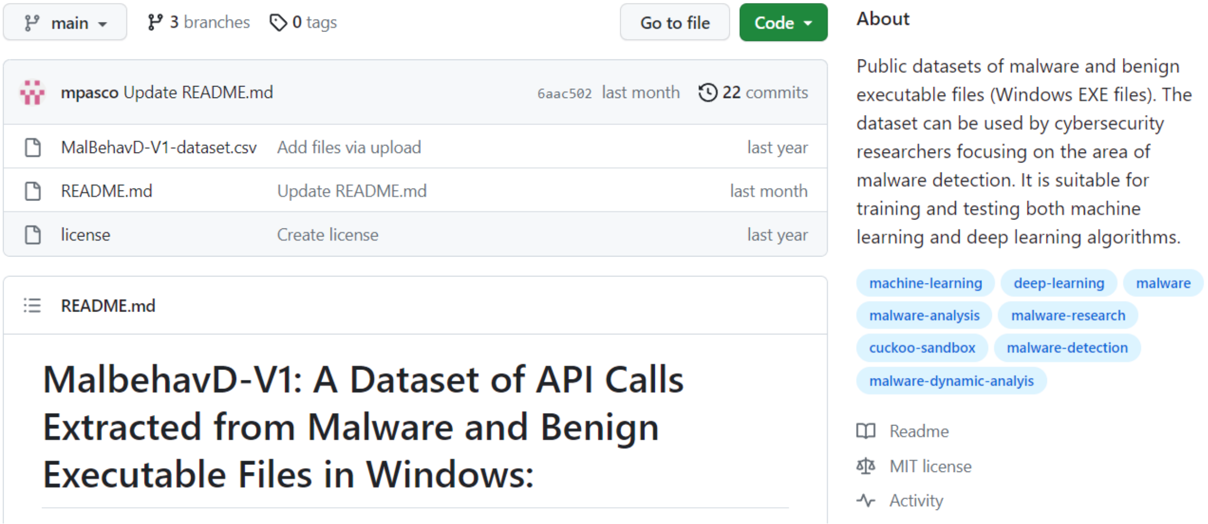
Dataset

Dataset Title	Dataset Alt. Title	Reference
MalbehavD-V1	MalbehavD-V1	[3]
Malware Analysis Datasets: API Call Sequences	Oliveira	[4]
Windows Malware Dataset with PE API Calls	Catak	[5]

Dataset - Dataset Exploration

MalbehavD-V1

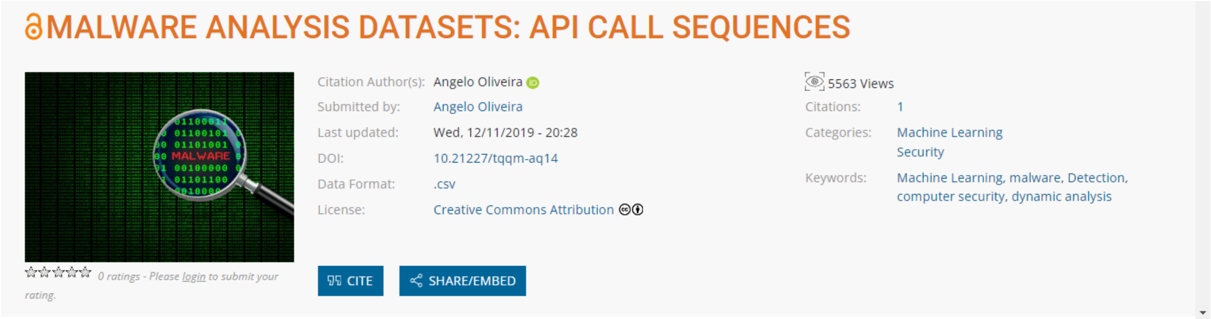
MalbehavD-V1 is found in this link which is a repository containing a lone CSV file which is the dataset itself.



Dataset - Dataset Exploration

Oliveira

Oliveira/'Malware Analysis Datasets: API Call Sequences' is found in this link which is a file (possibly downloaded as a ZIP) containing a lone CSV file which is the dataset itself.

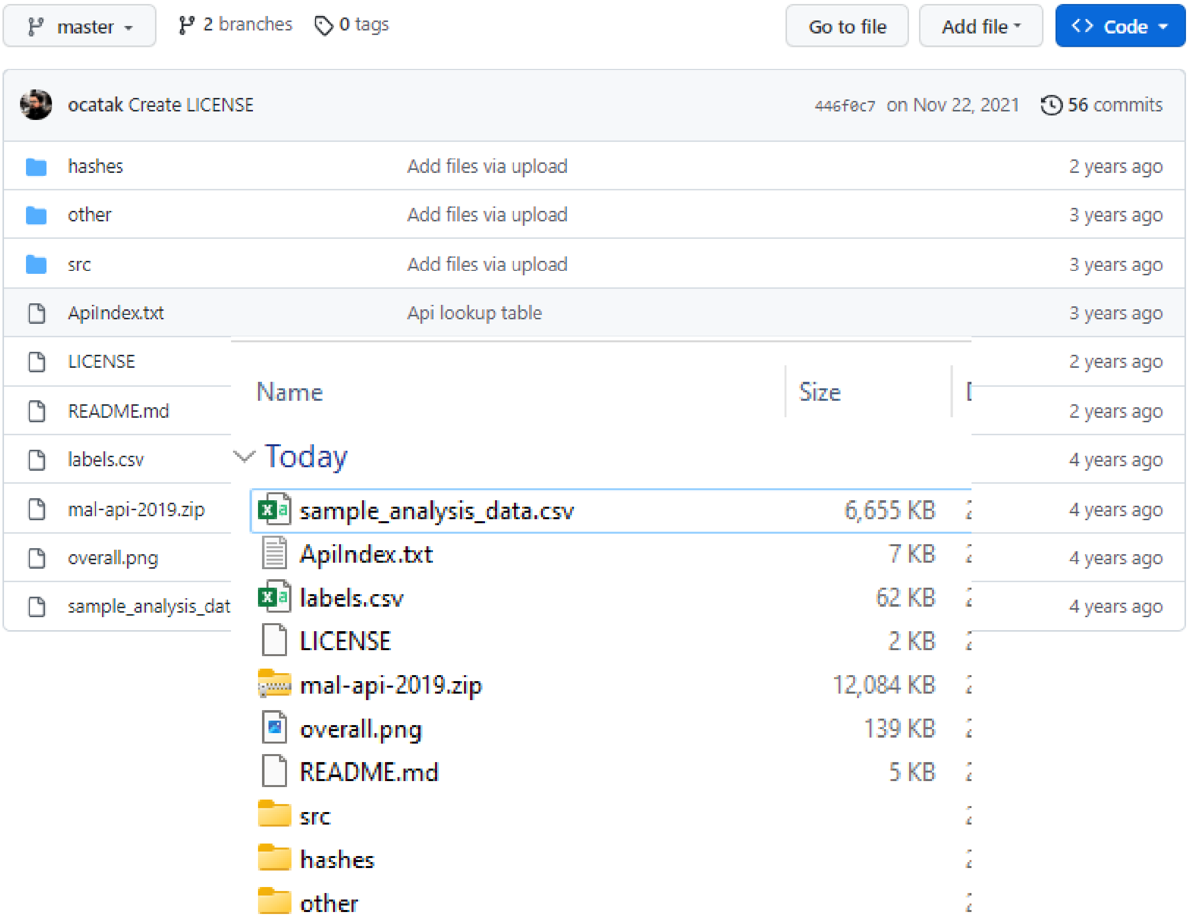


Dataset - Dataset Exploration

Catak

Catak/’Windows Malware Dataset with PE API Calls’ is found in this [link](#) which is a repository containing various files.

The zipped file ‘mal-api-2019.zip’, when extracted, contains a 2.01GB file called ‘all_analysis_data.csv’ where each row is linked to the ‘labels.csv’.



Dataset - Pre-Cleaning Procedure

The dataset cleaning procedure for each dataset will follow the steps mentioned below. MalbehavD-V1 will be the standard/reference ‘cleaned’ dataset where other datasets must follow suit.

1. **Oliveira**
   a. Making a version that contains string API calls.
   b. Removing ‘t_x’ prefix in API call column headers (i.e., only consecutive numbers).
   c. Moving position of ‘malware’ column to the second column.
2. **Catak**
   a. Replacing spaces to commas as delimiters for each API call.
   b. Changing features/API string casing from lower-case to camel case.
   c. Removing repeated API calls per row/sample (i.e., only first instance will be retained).
   d. Adding a first column for ‘malware_types’ where the per-row contents will be from the contents of ‘labels.csv’
   e. Adding a second column for ‘malware’ which will contain 1s.
3. **All Datasets**
   a. The NaN values (as seen in Pandas) will be replaced/represente with a space character (‘ ’) instead as LabelEncoding does not support NaN values.

Dataset - Pre-Cleaning Results

	sha256	malware	0	1	2	3	4	5	6
0	5c18291c481a192ed5003084dab2d8a117fd3736359218...	0	LdrUnloadDll	CoUninitialize	NtQueryKey	NtDuplicateObject	GetShortPathNameW	GetSystemInfo	IsDebuggerPresent
1	4683faf3da550ffb594cf5513c4cbb34f64df85f27fd1c...	0	NtOpenMutant	GetForegroundWindow	NtQueryKey	DrawTextExW	NtSetInformationFile	RegQueryValueExA	LdrGetProcedureAddress
2	9a0aea1c7290031d7c3429d0e921f107282cc6eab854ee...	0	GetForegroundWindow	DrawTextExW	GetSystemInfo	IsDebuggerPresent	GetSystemWindowsDirectoryW	NtQueryValueKey	RegCloseKey
3	e0f3e4d5f50afd9c31e51dd9941c5a52d57c7c524f5d11...	0	NtQueryValueKey	LdrUnloadDll	GlobalMemoryStatus	WriteConsoleA	NtOpenKey	LdrGetProcedureAddress	NtTerminateProcess
4	ec2b6d29992f13e74015ff0b129150b4afae15c593e4b7...	0	LdrUnloadDll	GetSystemTimeAsFileTime	NtOpenKey	WSAStartup	SetUnhandledExceptionFilter	NtTerminateProcess	NtClose

5 rows × 177 columns

Pre-Cleaned MalbehavD-V1

	hash	malware	0	1	2	3	4	5	6	7	...	
0	071e8c3f8922e186e57548cd4c703a5d	1	HttpSendRequestA	WSAAccept	NtCreateSection	Process32NextW	WSAAccept	NtCreateSection	Process32NextW	recvfrom	...	Internet(
1	33f8e6d08a6aae939f25a8e0d63dd523	1	GetFileVersionInfoExW	OleInitialize	NtQueryKey	OleInitialize	NtLoadKey	InternetConnectA	NtLoadKey	InternetConnectA	...	FindRe
2	b68abd064e975e1c6d5f25e748663076	1	CreateActCtxW	HttpOpenRequestW	RemoveDirectoryW	InternetConnectA	RemoveDirectoryW	InternetConnectA	RemoveDirectoryW	InternetConnectA	...	InternetGetConnected
3	72049be7bd30ea61297ea624ae198067	1	GetFileVersionInfoExW	OleInitialize	NtQueryKey	OleInitialize	NtLoadKey	InternetConnectA	NtLoadKey	InternetConnectA	...	Process
4	c9b3700a77facf29172f32df6bc77f48	1	GetFileVersionInfoExW	RemoveDirectoryW	InternetConnectA	RemoveDirectoryW	InternetConnectA	RemoveDirectoryW	InternetConnectA	RemoveDirectoryW	...	CryptUnprotec

5 rows × 102 columns

Pre-Cleaned Oliveira

	malware_type	malware	0	1	2	3	4	5	6	7	...	156	157	158	15
0	Trojan	1	LdrLoadDll	LdrGetProcedureAddress	RegOpenKeyExA	NtOpenKey	NtOpenKeyEx	NtQueryValueKey	NtClose	NtQueryAttributesFile	...	NaN	NaN	NaN	Na
1	Trojan	1	GetSystemTimeAsFileTime	NtAllocateVirtualMemory	NtFreeVirtualMemory	LdrGetDllHandle	LdrGetProcedureAddress	SetUnhandledExceptionFilter	NtCreateMutant	NtClose	...	NaN	NaN	NaN	Na
2	Backdoor	1	LdrGetDllHandle	LdrGetProcedureAddress	GetSystemDirectoryA	CopyFileA	RegOpenKeyExA	RegSetValueExA	RegCloseKey	RegCreateKeyExA	...	NaN	NaN	NaN	Na
3	Backdoor	1	LdrLoadDll	LdrGetProcedureAddress	RegOpenKeyExA	NtOpenKey	NtOpenKeyEx	NtQueryValueKey	NtClose	NtQueryAttributesFile	...	NaN	NaN	NaN	Na
4	Trojan	1	LdrLoadDll	LdrGetProcedureAddress	WSAStartup	NtCreateMutant	RegOpenKeyExA	RegDeleteKeyA	RegCloseKey	CopyFileA	...	NaN	NaN	NaN	Na

5 rows × 168 columns

Pre-Cleaned Catak

Dataset - Exploring Pre-Cleaned Data

Oliveira		MalbehavD-V1		Quantity	
API Call	**Size**	**API Call**	**Size**	**API Call**	**Size**
InternetConnectA	732701	NtClose	2524	NtOpenKeyEx	7686
RemoveDirectoryW	412278	NtQueryValueKey	2447	NtClose	7030
NtLoadKey	314298	NtOpenKey	2446	NtOpenKey	6656
SetStdHandle	260429	LdrGetProcedureAddress	2324	LdrGetProcedureAddress	6599
Process32NextW	222786	NtCreateFile	2181	NtQueryValueKey	6520
GetFileType	191735	NtAllocateVirtualMemory	2173	NtAllocateVirtualMemory	6449
GetAdaptersAddresses	188556	LdrUnloadDll	2139	LdrLoadDll	6408
FindResourceW	186112	RegCloseKey	1936	LdrGetDllHandle	6201
LookupAccountSidW	184347	LdrGetDllHandle	1922	NtCreateFile	5969
OleInitialize	176499	LdrLoadDll	1894	NtFreeVirtualMemory	5545
WSAAccept	102230	NtFreeVirtualMemory	1891	RegCloseKey	5515
NtCreateSection	101969	GetSystemTimeAsFileTime	1796	RegOpenKeyExA	5395
CryptHashData	83484	NtReadFile	1596	LdrUnloadDll	5287
GetFileVersionInfoExW	72684	NtTerminateProcess	1544	NtMapViewOfSection	4373
__anomaly__	53122	NtMapViewOfSection	1539	RegOpenKeyExW	4239
recv	39452	NtCreateSection	1473	NtTerminateProcess	4160
NtQueryKey	39237	NtWriteFile	1455	RegQueryValueExA	4051
NtSetValueKey	37756	RegOpenKeyExW	1433	NtCreateSection	3946
HttpSendRequestA	33689	RegQueryValueExW	1339	RegQueryValueExW	3883
RegEnumValueA	32925	GetFileAttributesW	1338	GetSystemMetrics	3688

Dataset - Labels

```
y = malbehavd['malware'].to_numpy()
labels = malbehavd['malware'].unique()
print("MalbehavD - No. of unique labels: ", labels.size)
print(labels)
```

```
MalbehavD - No. of unique labels:  2
[0 1]
```

```
y = oliveira['malware'].to_numpy()
labels = oliveira['malware'].unique()
print("Oliviera - No. of unique labels: ", labels.size)
print(labels)
```

```
Oliviera - No. of unique labels:  2
[1 0]
```

```
y = catak['malware'].to_numpy()
labels = catak['malware'].unique()
print("Catak - No. of unique labels: ", labels.size)
print(labels)
```

```
Catak - No. of unique labels:  1
[1]
```

Dataset - API Call Matching (API Call Coverage)

Dataset	API Match/Coverage
MalbehavD-V1	94.48%
Oliveira	85.39%
Catak*	92.53%

*Irrelevant due to intent/purpose/nature of the dataset

Dataset – Similarity amongst Datasets

Dataset Pair	Match/Similarity Rate
MalbehavD-V1 to Oliveira	85.5670%
MalbehavD-V1 to Catak*	93.4708%

*Irrelevant due to intent/purpose/nature of the dataset

Dataset - Maximizing and Trimming (for Robustness Testing)

- **The no. of features used in the fitted dataset must also be the same as the test/input dataset.** Hence, another dataset pre-processing technique must be done to allow for cross-dataset training and testing. The datasets will then be LabelEncoded once re-shaped.
- ‘Expansion/Maximizing’
  - The datasets will be **resized to the biggest dataset in terms of column size** which 175 columns (including the 2 labels) from MalbehavD-V1. Extending the datasets will result to adding NaN values which will be represented by a space character.
- ‘Trimming/Minimizing’
  - The datasets will be **resized to the smallest dataset in terms of column size** which is 102 columns (including the 2 labels) from the Oliveira dataset. Trimming the datasets will result to the excess API calls to be trimmed.

Dataset - LabelEncoding

- Building ML models requires datasets that contain numeric feature data, hence there must be a way to convert the string API calls to numeric form such that a number equates to a specific API call which is found in [4]
- Effectively, this is ‘feature conversion’.
- Among the techniques that can be used in SciKit Learn is LabelEncoding or OneHotEncoder.
- The list of APIs used as reference came from ‘CombinedAPIs.csv’ where it will be used on the LabelEncoder pre-processing function.

Dataset - LabelEncoding

	0	1	2	3	4	5
0	LdrUnloadDll	CoUninitialize	NtQueryKey	NtDuplicateObject	GetShortPathNameW	GetSystemInfo
1	NtOpenMutant	GetForegroundWindow	NtQueryKey	DrawTextExW	NtSetInformationFile	RegQueryValueExA
2	GetForegroundWindow	DrawTextExW	GetSystemInfo	IsDebuggerPresent	GetSystemWindowsDirectoryW	NtQueryValueKey
3	NtQueryValueKey	LdrUnloadDll	GlobalMemoryStatus	WriteConsoleA	NtOpenKey	LdrGetProcedureAddress
4	LdrUnloadDll	GetSystemTimeAsFileTime	NtOpenKey	WSAStartup	SetUnhandledExceptionFilter	NtTerminateProcess

Before Label Encoding (MalbehavD-V1)

	0	1	2	3	4	5
0	26	2	65	52	29	31
1	35	19	65	8	78	92
2	14	7	26	35	34	77
3	39	31	35	95	66	43
4	26	23	60	94	104	83

After Label Encoding (MalbehavD-V1)

Dataset – Cleaned Dataset

Dataset (Filename)	Shape (Row x Col)	Element Size/Count	File Size – String	File Size - LabelEncoded
MalbehavD-V1	2570 x 177	454,890	2.55 MB	1.13 MB
Oliveira	43876 x 102	4,475,352	65.8 MB	15.4 MB
Catak	7106 x 168	1,193,808	6.73 MB	2.73 MB

Dataset – Cleaned Dataset Preview

2.1.3. Preview Catak

```
catak.head()
```

	malware_type	malware	0	1	2	3	4	5	6	7	...	156	157	158	159	160	161	162	163	164	165
0	Trojan	1	27	32	87	66	71	84	64	75	...	0	0	0	0	0	0	0	0	0	0
1	Trojan	1	19	39	54	41	45	118	67	54	...	0	0	0	0	0	0	0	0	0	0
2	Backdoor	1	25	32	22	2	99	109	106	96	...	0	0	0	0	0	0	0	0	0	0
3	Backdoor	1	27	32	87	66	71	84	64	75	...	0	0	0	0	0	0	0	0	0	0
4	Trojan	1	27	32	106	55	99	101	106	4	...	0	0	0	0	0	0	0	0	0	0

5 rows × 168 columns

Preview of Cleaned Catak (LabelEncoded)

Dataset – Cleaned Dataset Preview

2.1.1 Preview MalbehavD

```
malbehavd.head()
```

	sha256	malware	0	1	2	3	4	5	6	7	...	165	166	167	168	169	170	171	172	173	174
0	5c18291c481a192ed5003084dab2d8a117fd3736359218...	0	26	2	66	53	30	32	45	38	...	0	0	0	0	0	0	0	0	0	0
1	4683faf3da550ffb594cf5513c4cbb34f64df85f27fd1c...	0	35	19	66	10	79	93	47	2	...	0	0	0	0	0	0	0	0	0	0
2	9a0aea1c7290031d7c3429d0e921f107282cc6eab854ee...	0	14	7	27	36	35	78	95	23	...	0	0	0	0	0	0	0	0	0	0
3	e0f3e4d5f50afd9c31e51dd9941c5a52d57c7c524f5d11...	0	39	31	36	96	67	44	86	55	...	0	0	0	0	0	0	0	0	0	0
4	ec2b6d29992f13e74015ff0b129150b4afae15c593e4b7...	0	26	23	61	95	105	84	57	54	...	0	0	0	0	0	0	0	0	0	0

5 rows × 177 columns

Preview of Cleaned MalbehavD-V1 (LabelEncoded)

2.1.2. Preview Oliveira

```
oliveira.head()
```

	hash	malware	0	1	2	3	4	5	6	7	...	90	91	92	93	94	95	96	97	98	99
0	071e8c3f8922e186e57548cd4c703a5d	1	36	74	54	68	108	71	91	123	...	73	41	177	83	103	126	23	128	34	40
1	33f8e6d08a6aae939f25a8e0d63dd523	1	22	54	62	65	71	54	75	52	...	35	49	143	72	42	174	85	110	135	23
2	b68abd064e975e1c6d5f25e748663076	1	3	32	75	43	92	54	97	52	...	77	39	72	77	39	68	79	41	73	69
3	72049be7bd30ea61297ea624ae198067	1	22	54	62	65	71	54	75	52	...	129	122	180	126	182	109	128	185	141	177
4	c9b3700a77facf29172f32df6bc77f48	1	22	63	42	74	51	98	56	96	...	26	123	153	26	125	152	88	159	90	152

5 rows × 102 columns

Preview of Cleaned Oliveira (LabelEncoded)

Dataset - Pre-processing and Cleaning Time

7. Time Taken

```
dur_s = time.time()-start_time
dur_min = dur_s/60
print(f"{dur_s}s")
print(f"{dur_min:.2f}mins")
```

```
314.7926678657532s
5.25mins
```

ML Models

- Traditional
  - K-Nearest Neighbors (KNN)
  - Logistic Regression (LR)
  - Decision Tree (DT/DTC)
  - Support Vector Machine (SVM)
  - Random Forest (RF)
  - Gaussian Naive Bayes (GNB)
- **Boosted**
  - **AdaBoost**
  - **Gradient Tree Boosting (GBT)**
  - **Histogram-based Gradient Boosting Classification Tree (HGBT)**
- Neural Network Based
  - Multi-layer Perceptron (MLP)

ML Models Tuning

- The hyperparameters of a given ML/DL Model can be tuned which in turn can result to a better performing model.
- Tuning can be done manually or by means of ‘automated’ hyperparameter tuning.
- Due to limited time and computing resources, the hyperparameter tuning setup will be using ‘RandomizedSearchCV’ instead of the preferrable ‘GridSearchCV’. The hyperparameter setup for each model is also not that broad which can affect the quality of the ‘best’ tuning.

ML Models – Performance Metrics

- **Accuracy**
  - The ratio of cases the model correctly predicted.
- **F1-Score**
  - The harmonic mean of positive predictive value and recall.
  - Note: F1-Score Weighted was used
- **Precision**
  - The ratio of truly positive cases from all cases the model predicted positive.
- **Recall**
  - The ratio of positive cases predicted as positive.
- **ROC-AUC**
  - A range value (0-1; worst to perfect) representing the ROC curve.
  - ROC-AUC is graphed as the Recall/TPR on the y-axis and FPR on the x-axis [8].

ML Models – Other Metrics

Training Time (s) - The time taken by each model to train using a given dataset.

Prediction Time (ms) – The time taken by the model to predict a given sample.

Tuning Time (s) – The time taken by the model be automatically tuned using ‘RandomizedSearchCV’.

ML Models – Tests and Comparisons

Stratified K-Folds Test

On each dataset & configuration combination, the results will be based off the Stratified K-Fold Test to determine overfitting, especially during testing and tuning.

Default and Tuned Model Performance Comparison

The model's default and tuned config on each model trained on each dataset is compared to one another. The data to be presented in this test will be a range of
-/+ percentages where the [-] values will indicate that Default Model is a better tuning while a [+] value will indicate that Tuned Model is a better one.

ML Models – Tests and Comparisons

Dataset Performance Comparison

The dataset's impact in model performance (on both default and tuned) is compared to one another. The data to be presented in this test will be a range of -/+ percentages where the [-] values will indicate that MalbehavD-V1 is a better dataset while a [+] value will indicate that Oliveira is a better one.

Model Robustness Test

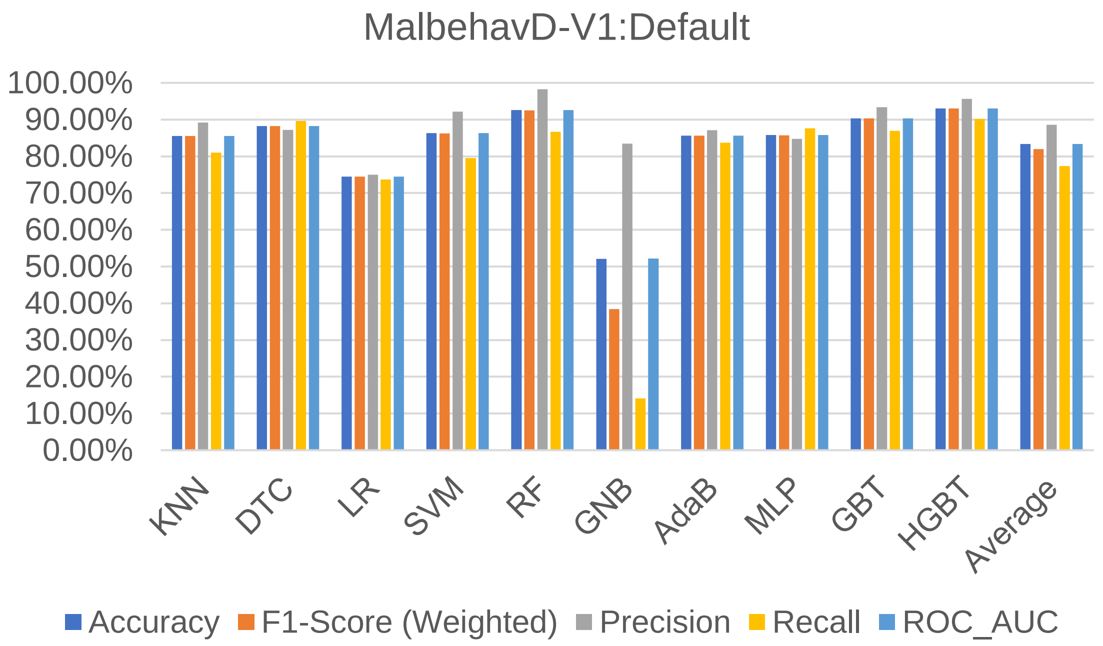
Model Robustness will be tested by means of training a model on one dataset and testing on another. This aims to determine how good a model is when encountering a similar dataset but not exactly as trained. This test also investigates which dataset reshaping technique (i.e., trimming/maximizing) performs better.

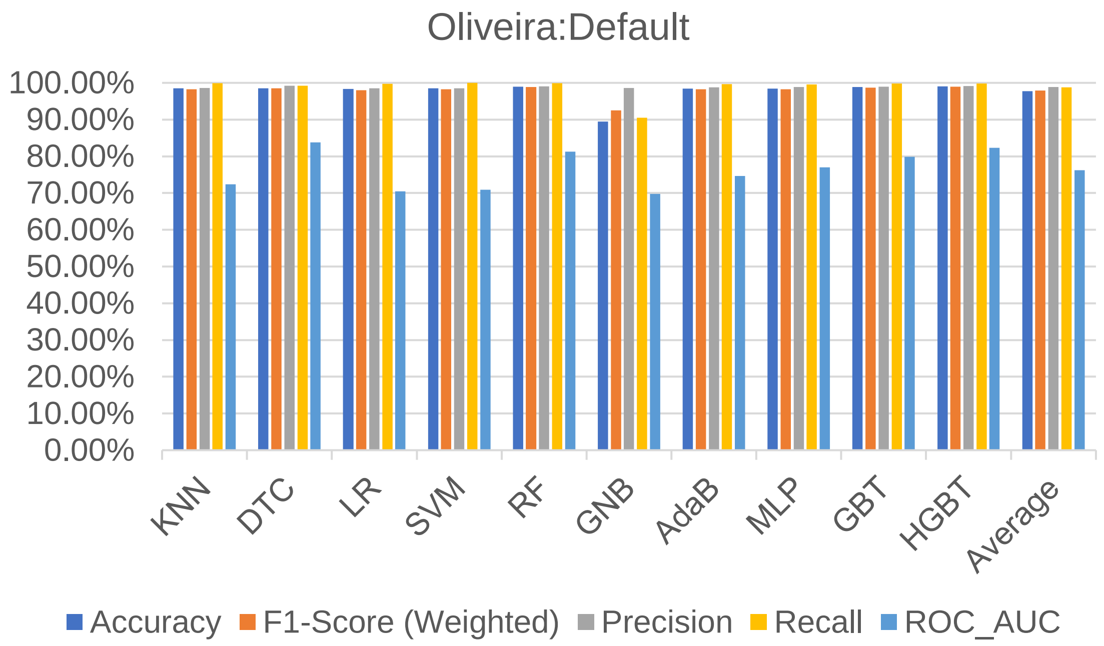
Time Test

Testing training and prediction time for each model and dataset combination. The increase in time is also measured here where [-] means a reduction in time while a [+] means an increase in time.

Results

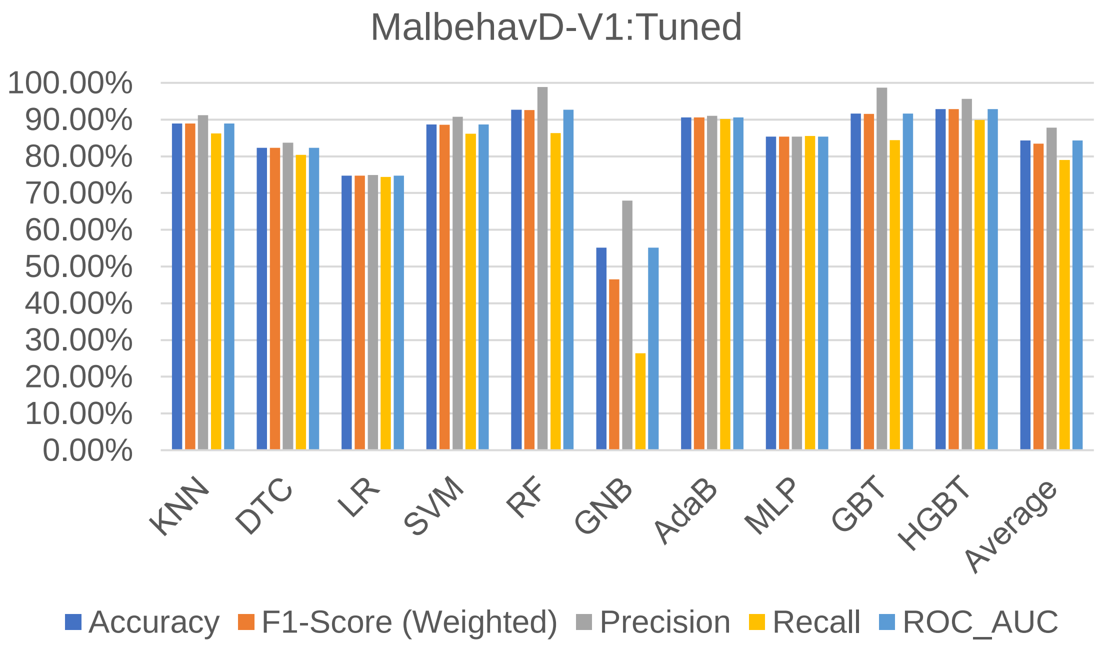
Default vs Tuned Model Performance Comparison

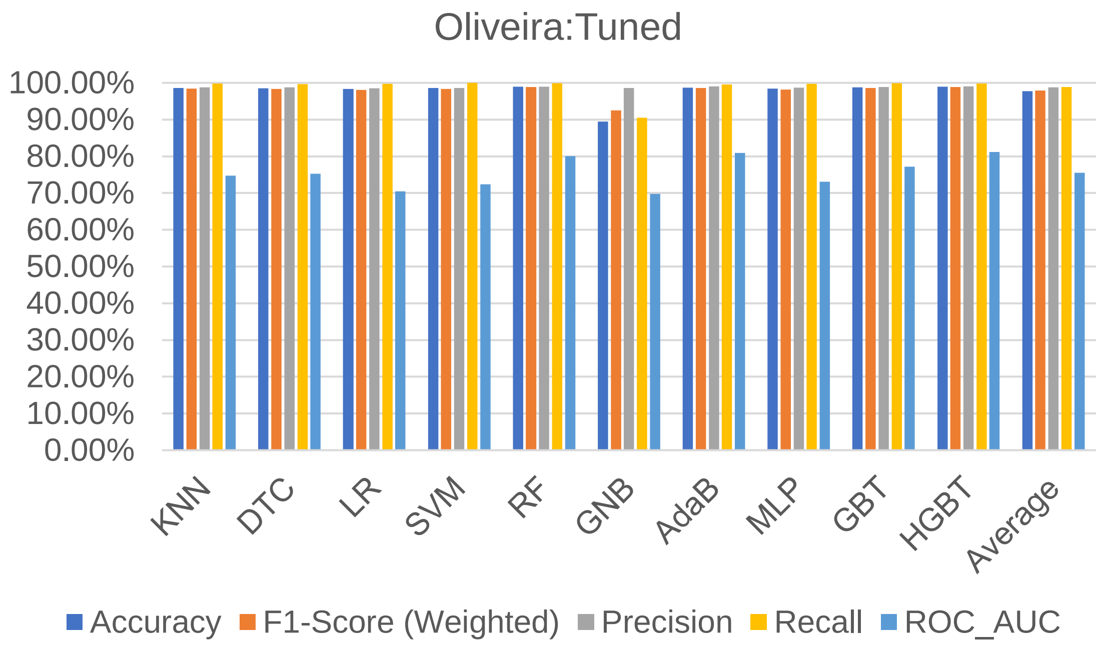




Results

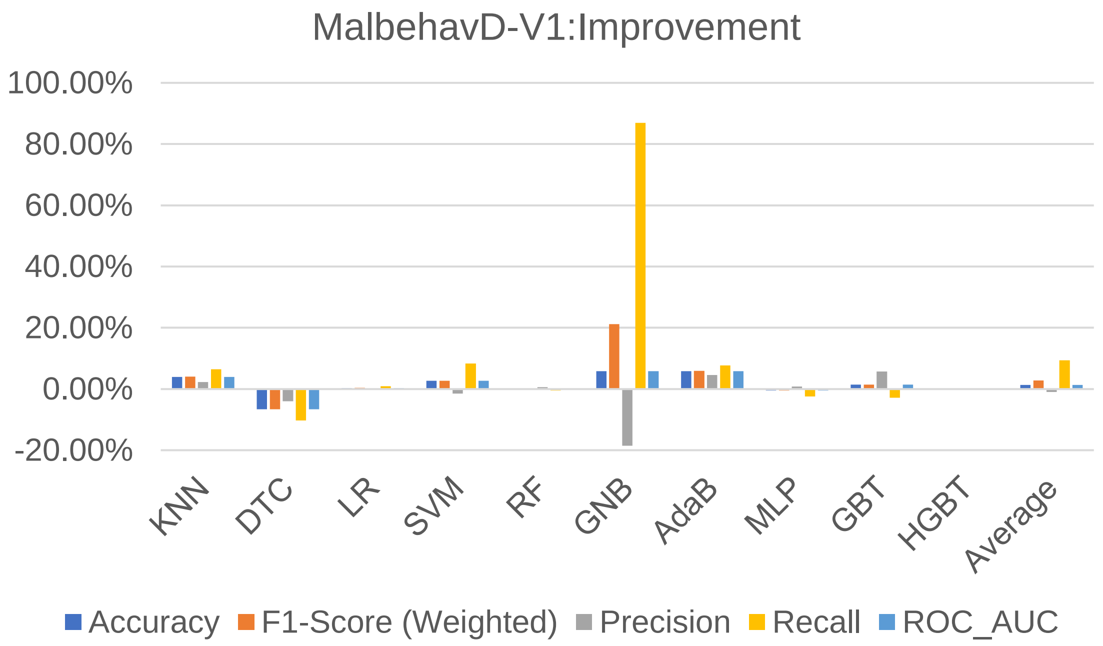
Default vs Tuned Model Performance Comparison

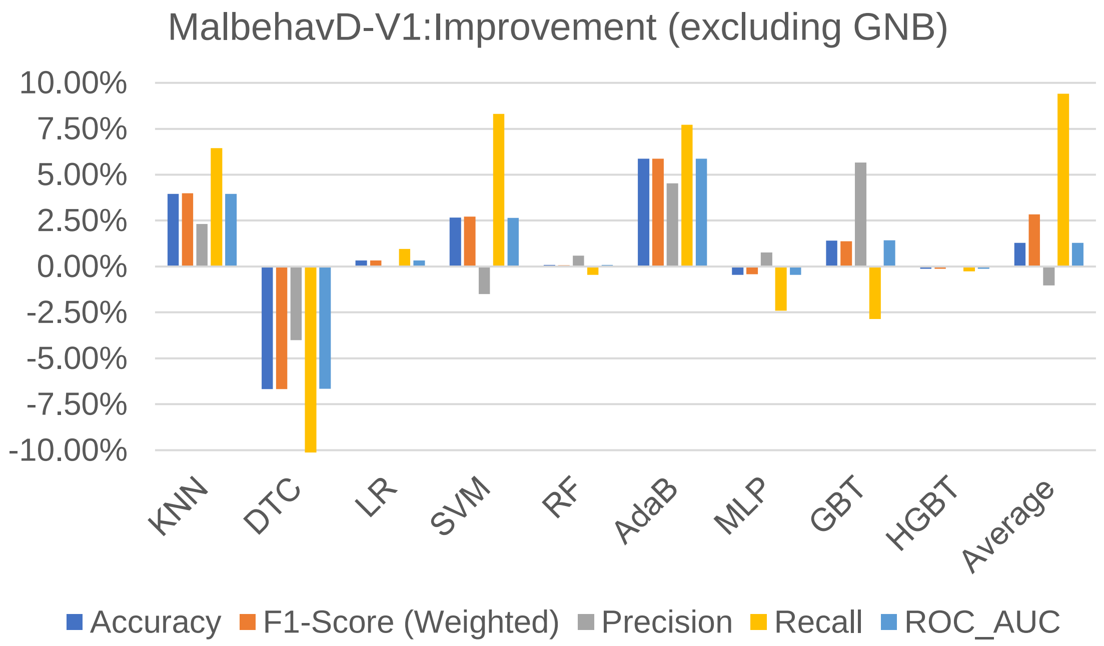




Results

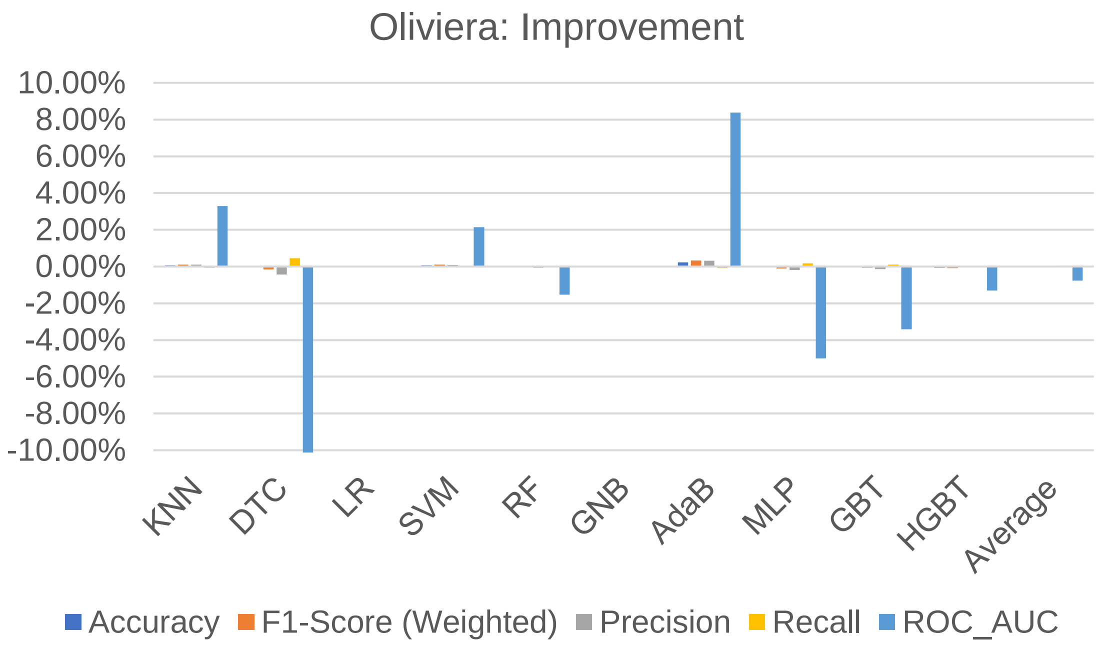
Default vs Tuned Model Performance Comparison

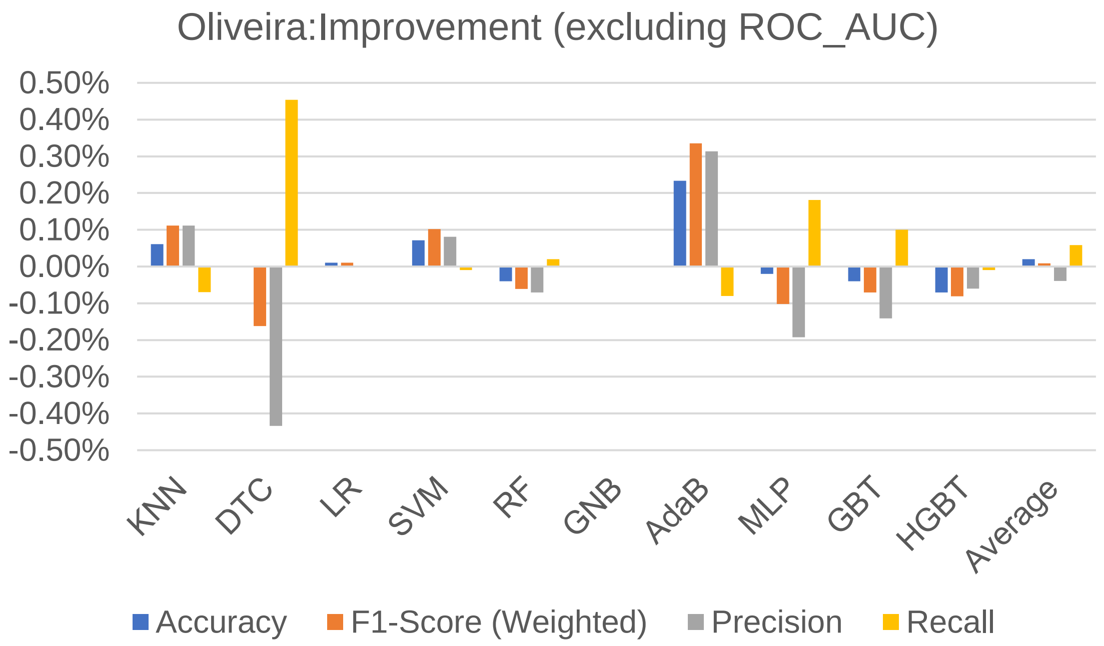




Results

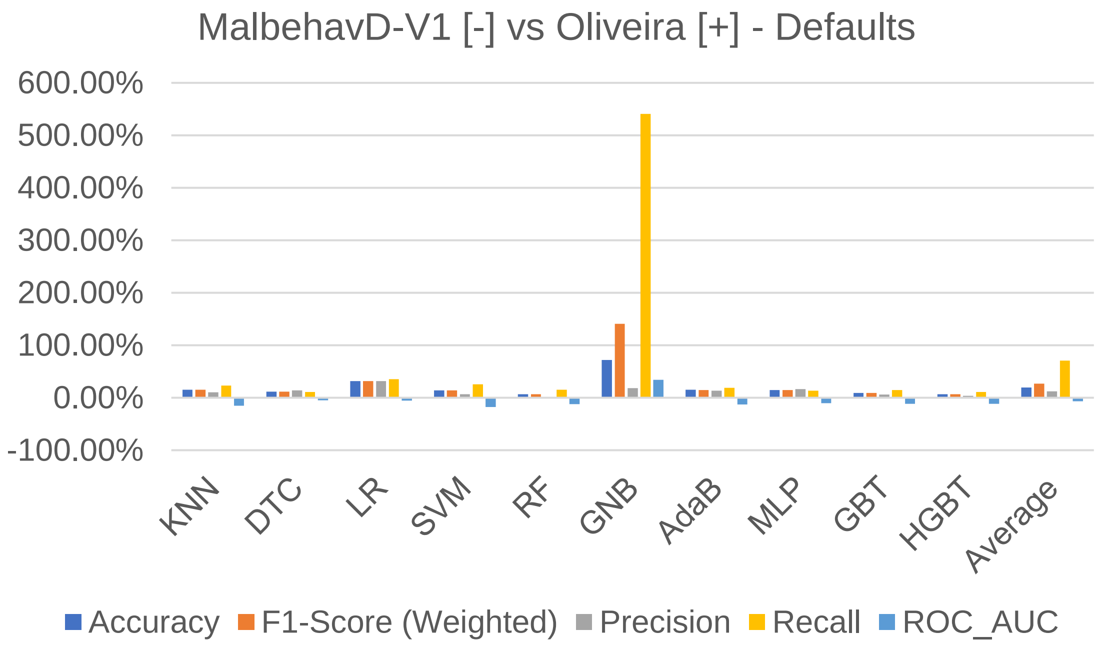
Default vs Tuned Model Performance Comparison

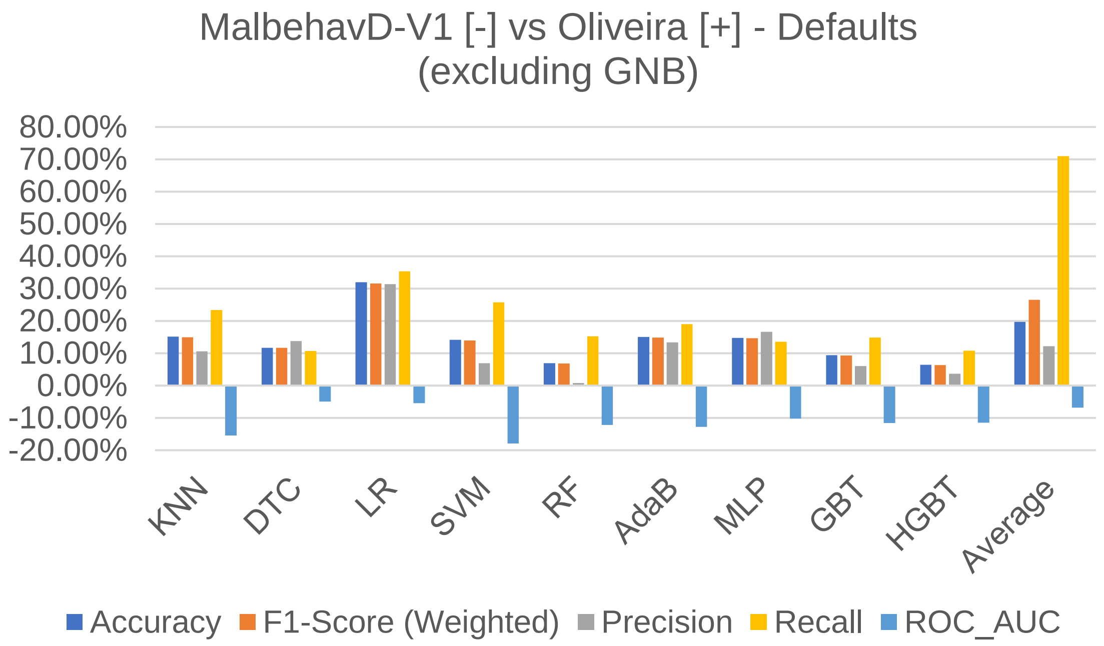




Results

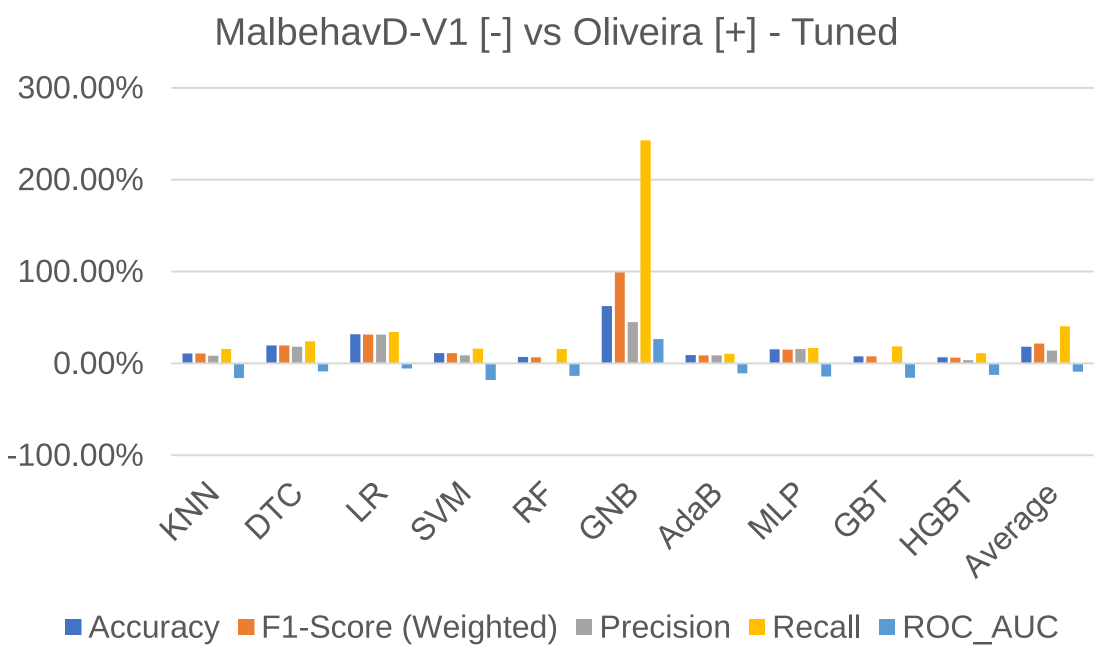
Dataset Performance Comparison

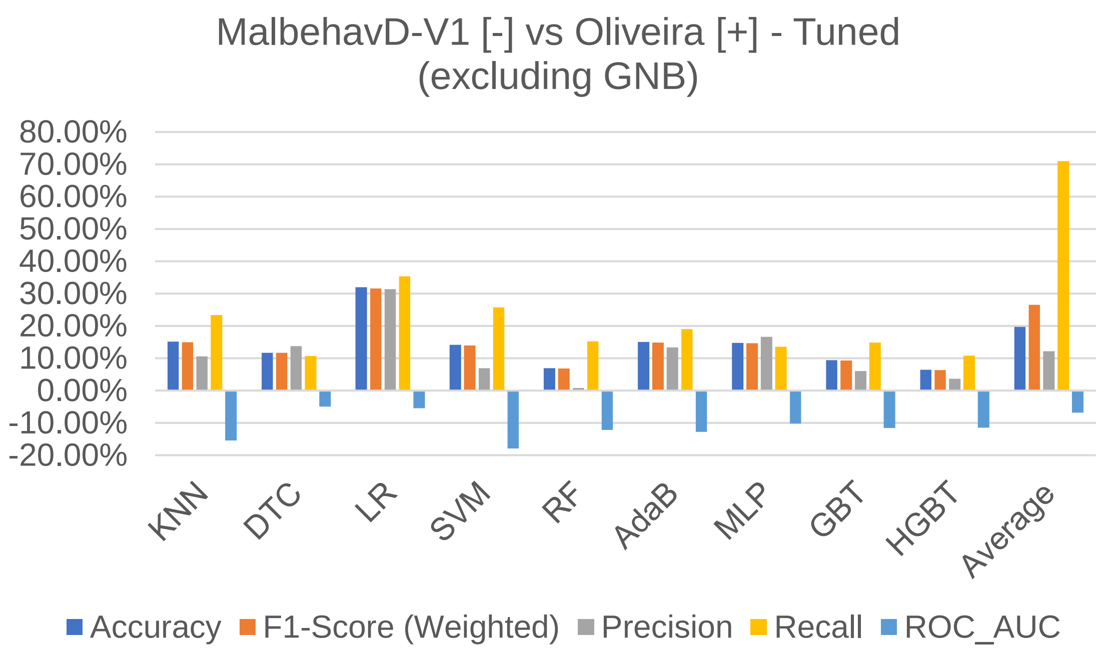




Results

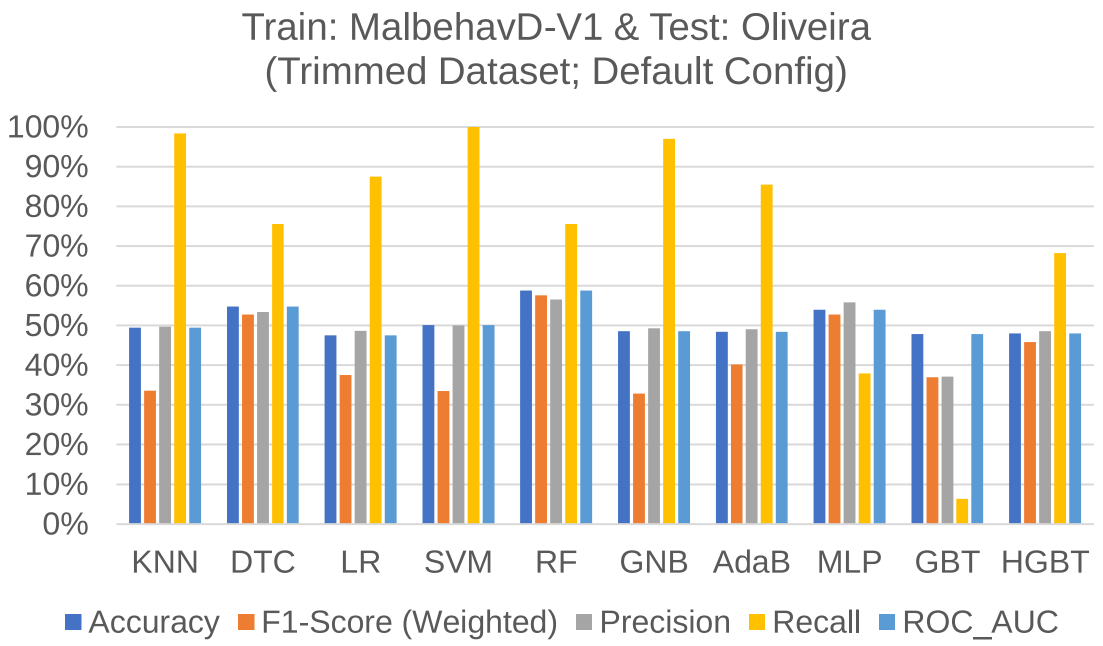
Dataset Performance Comparison

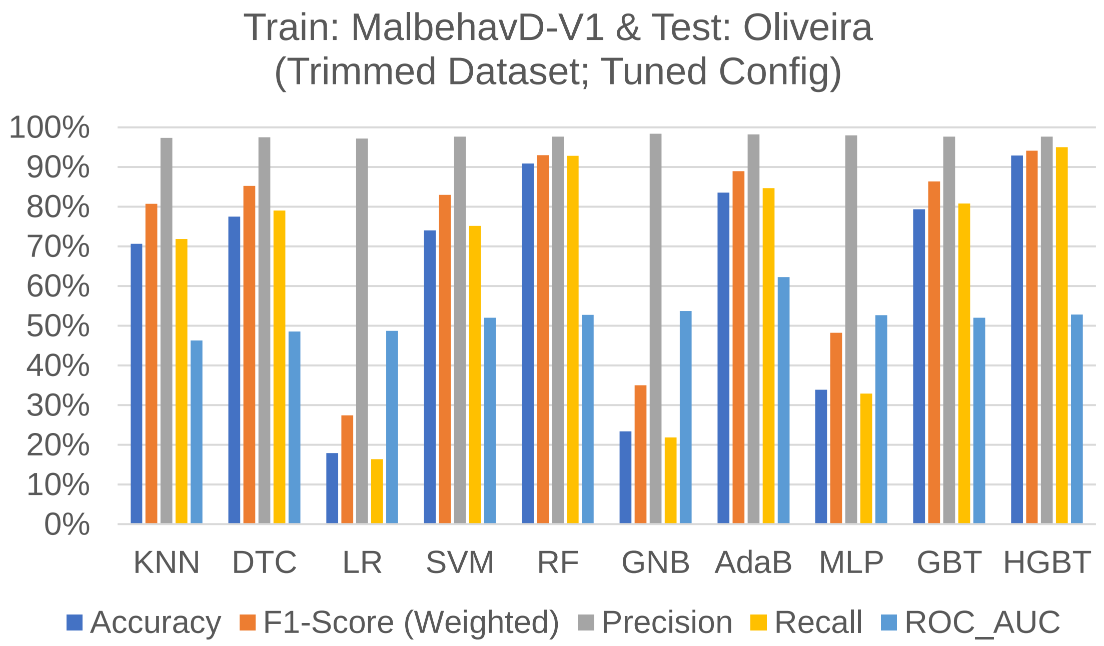




Results

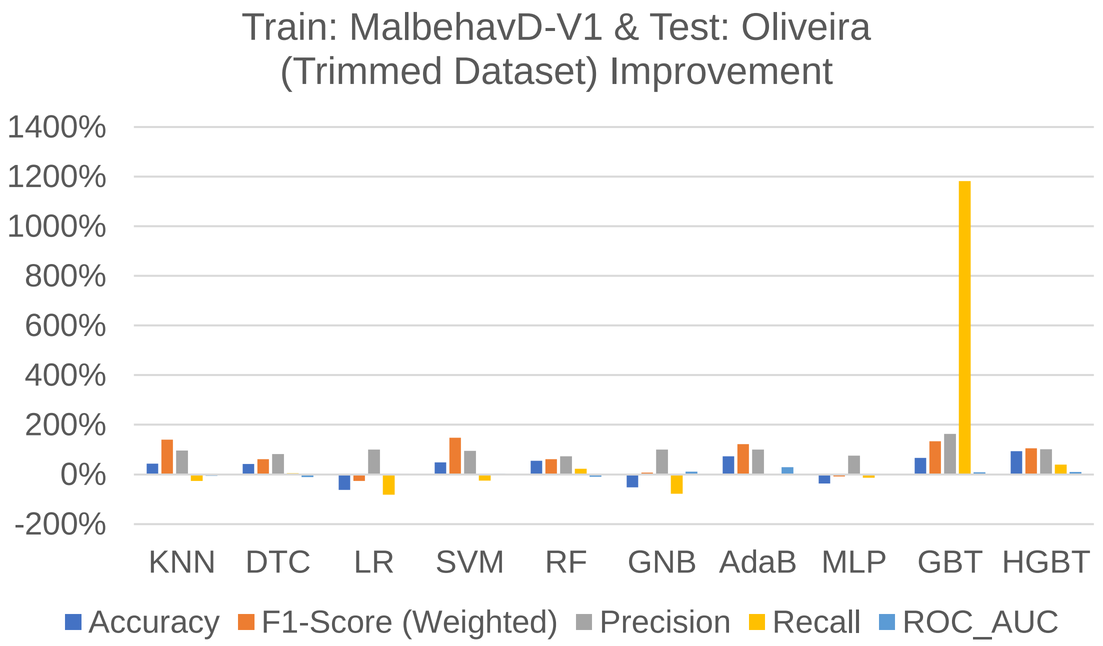
Model Robustness Test

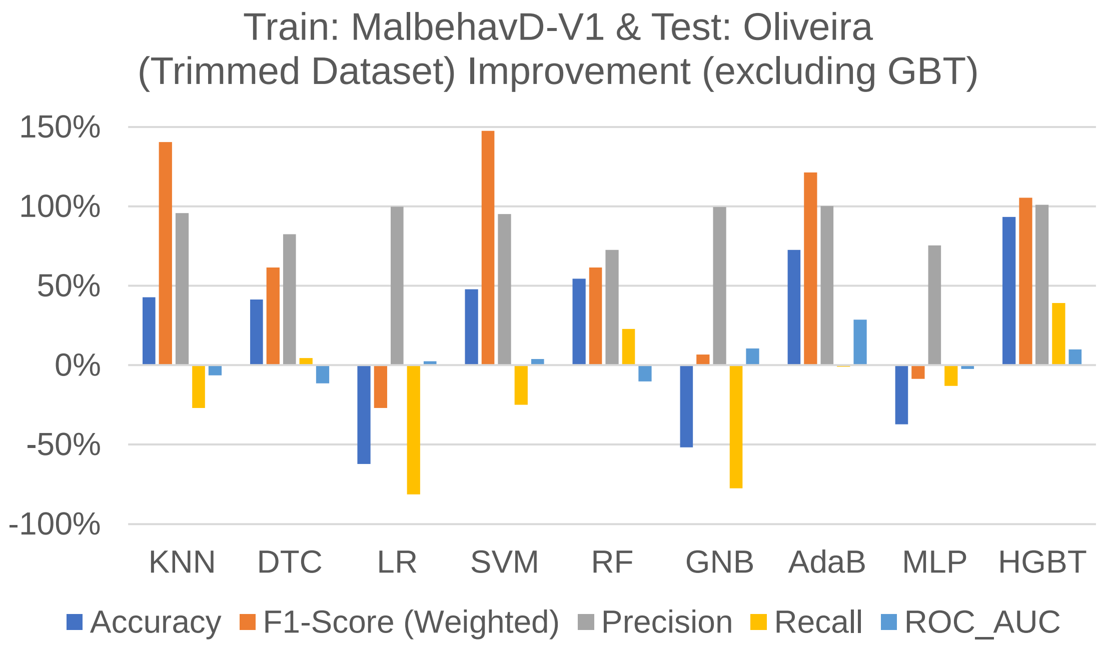




Results

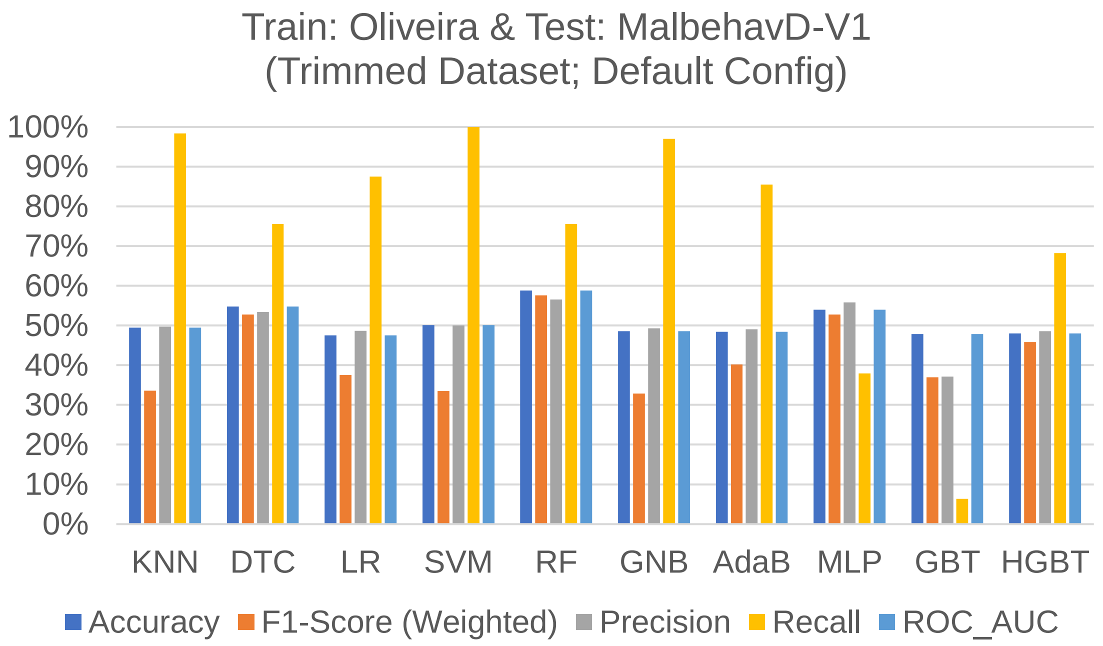
Model Robustness Test

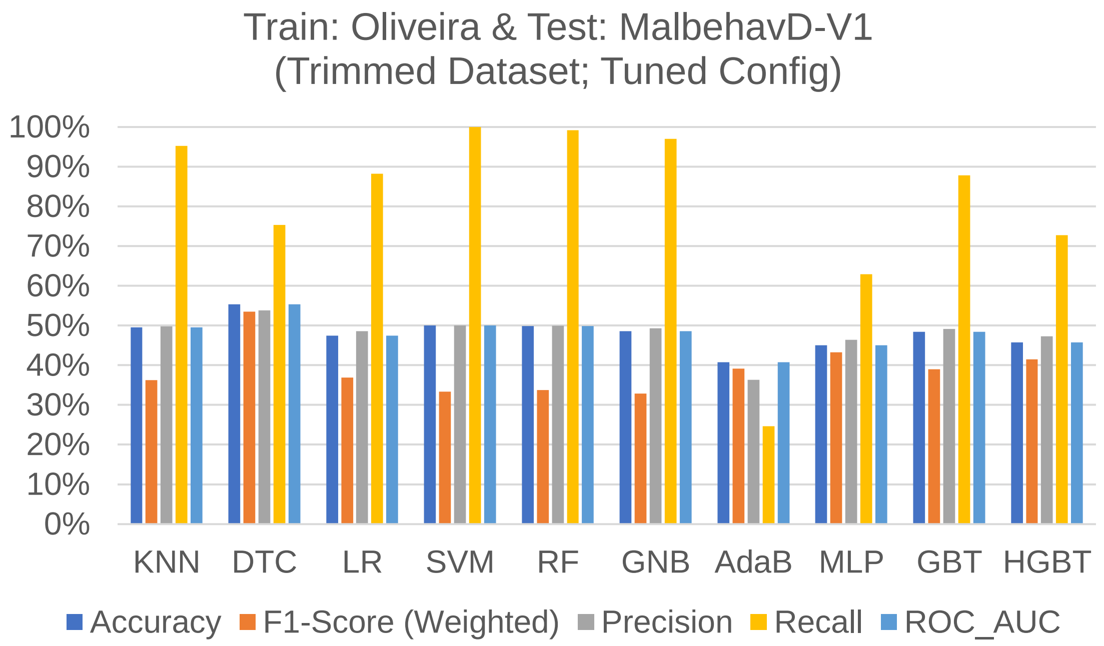




Results

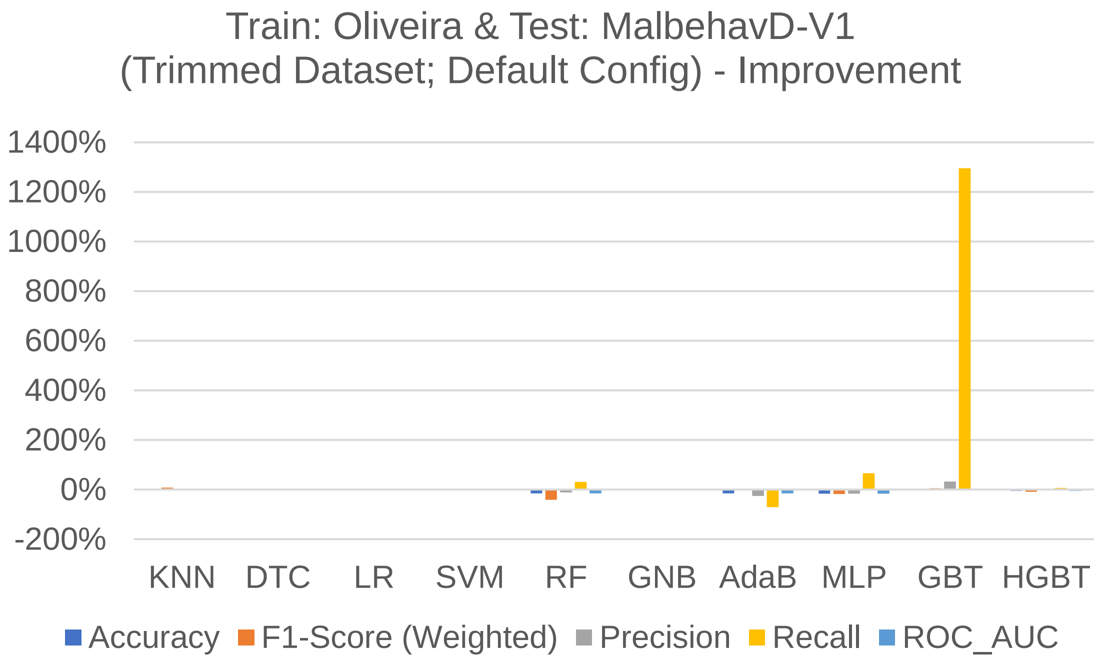
Model Robustness Test

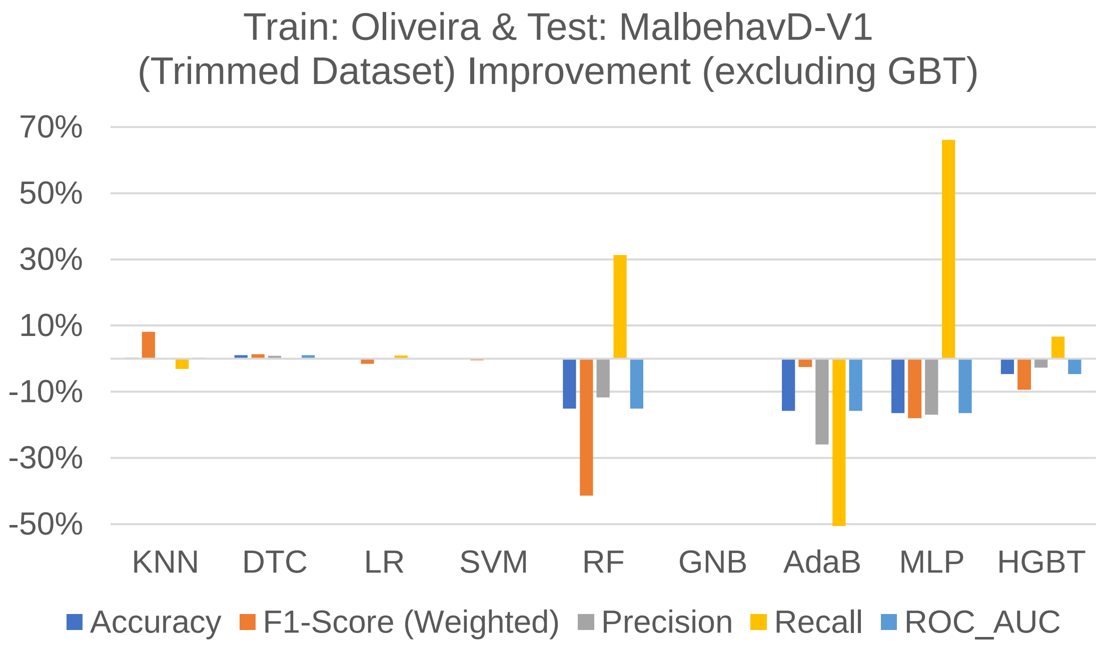




Results

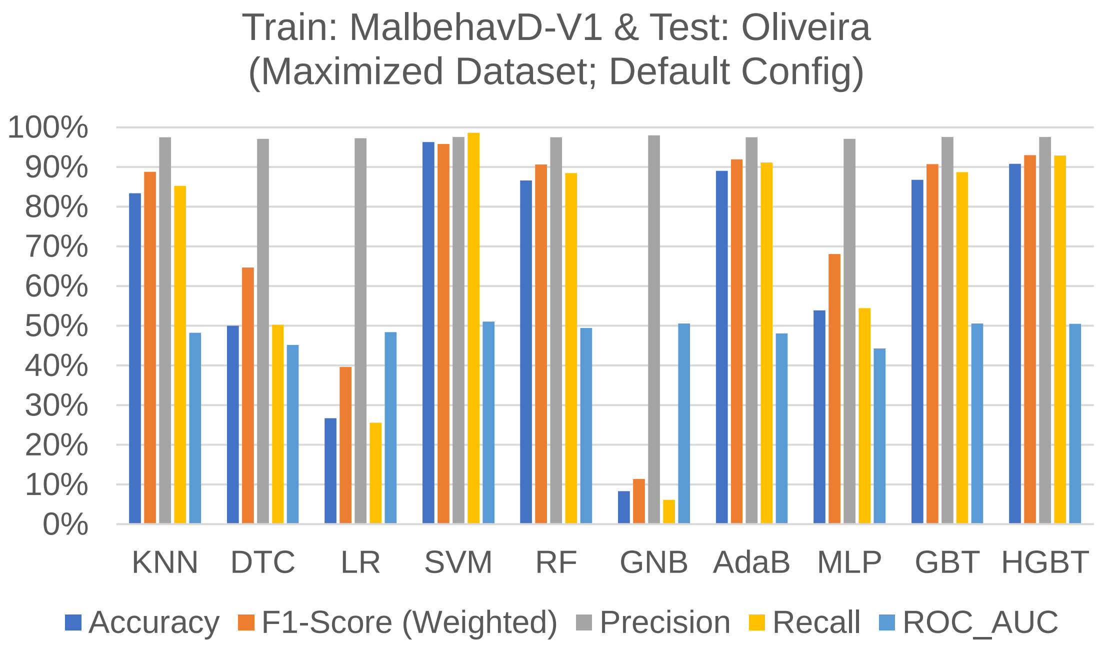
Model Robustness Test

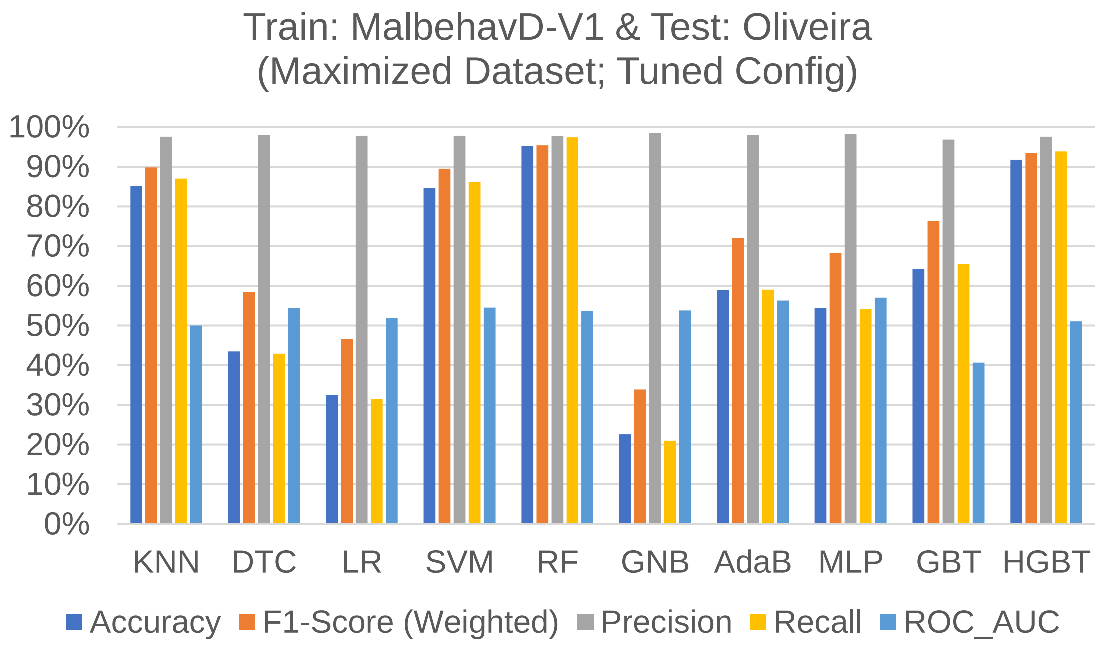




Results

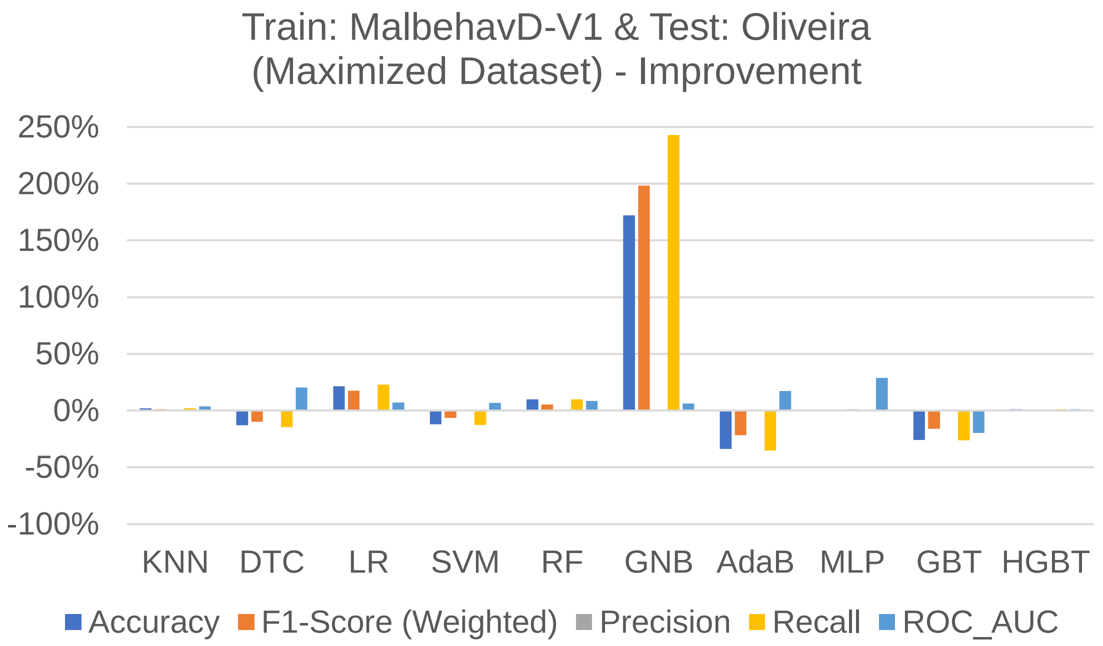
Model Robustness Test

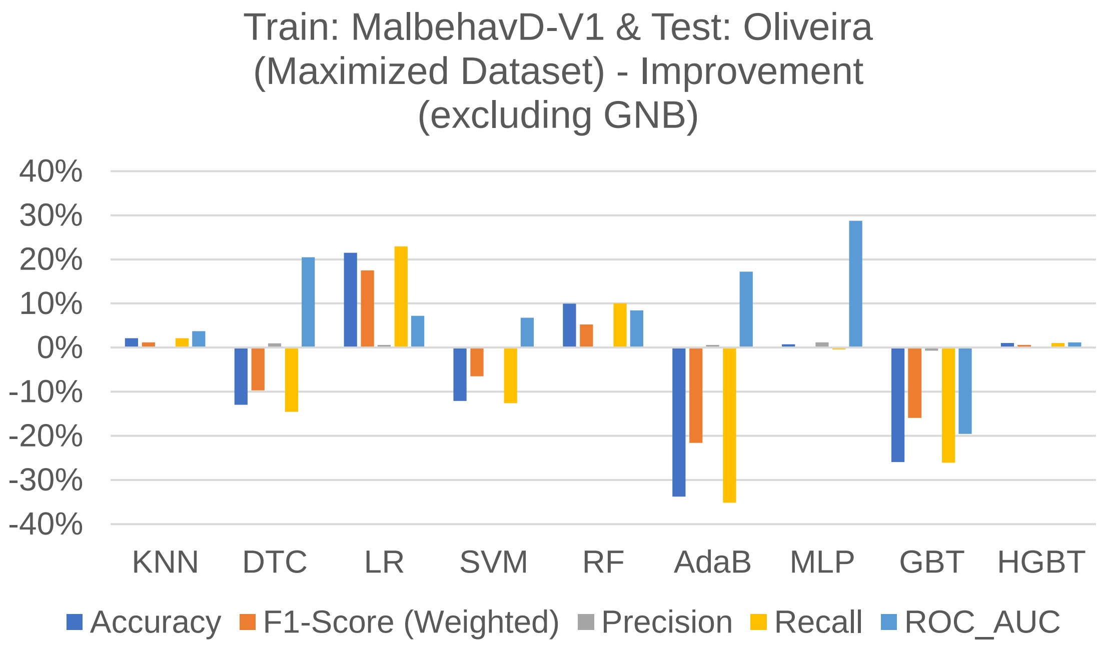




Results

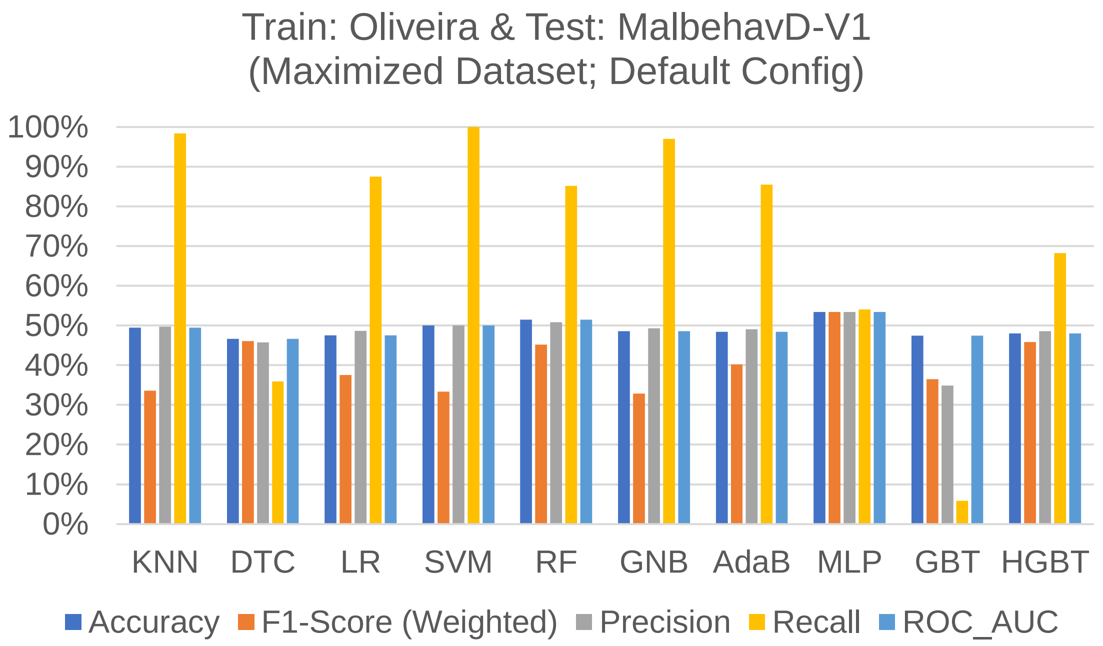
Model Robustness Test

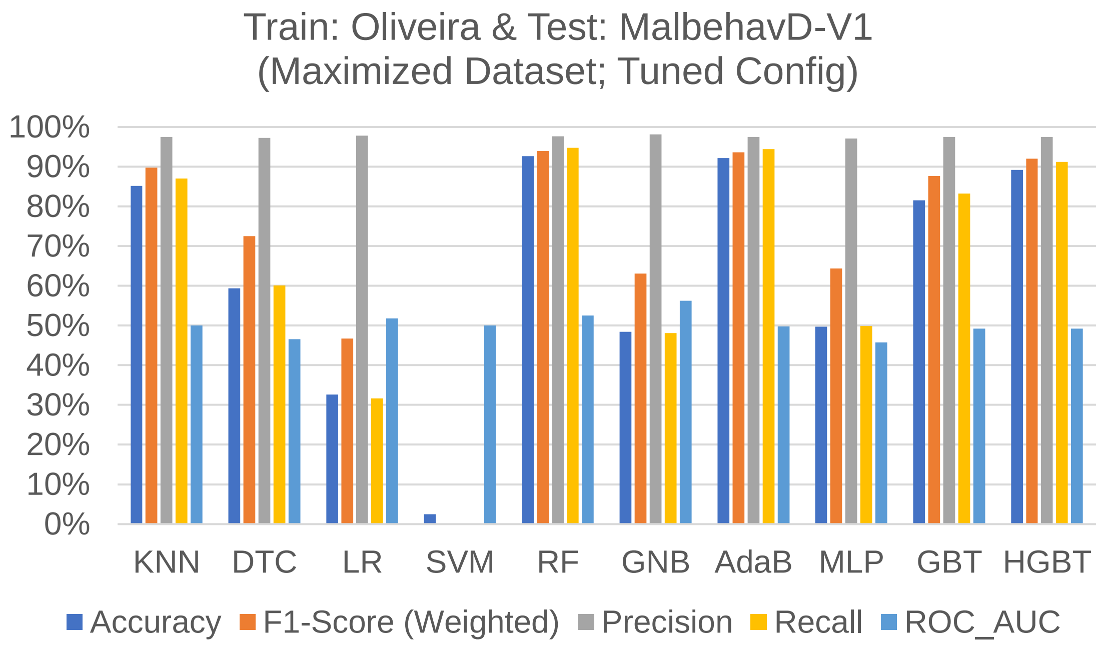




Results

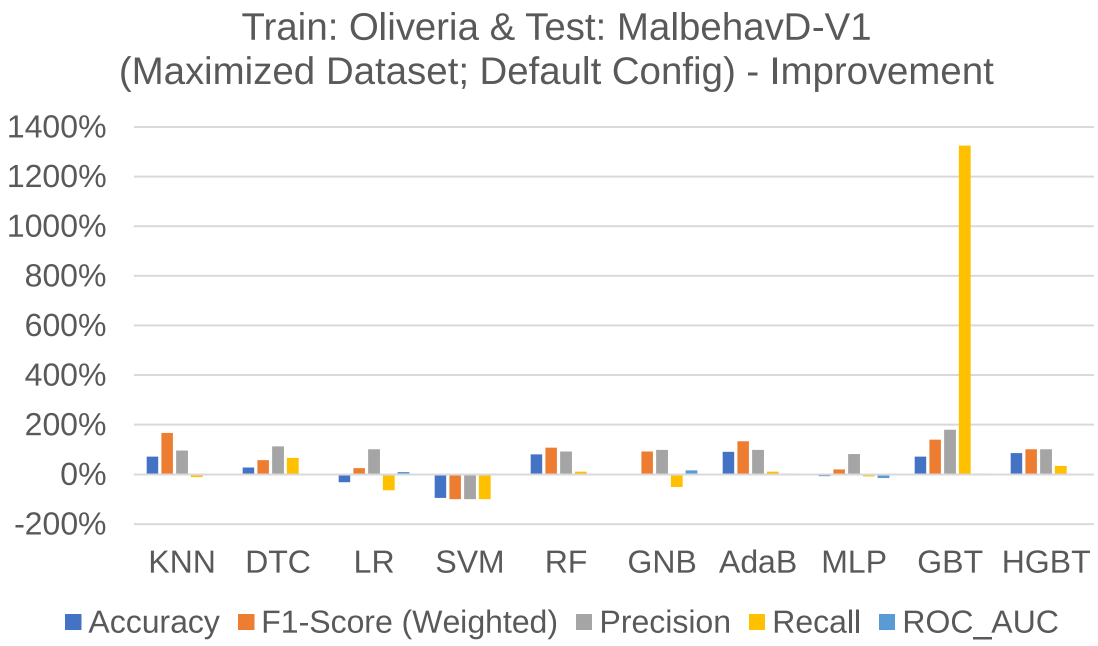
Model Robustness Test

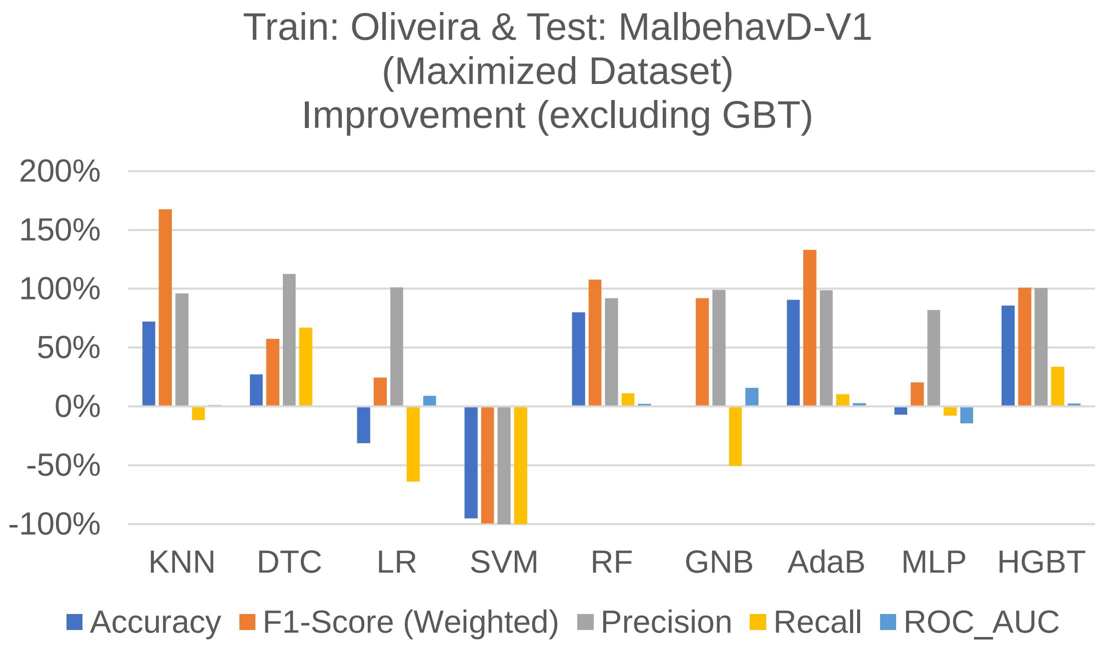




Results

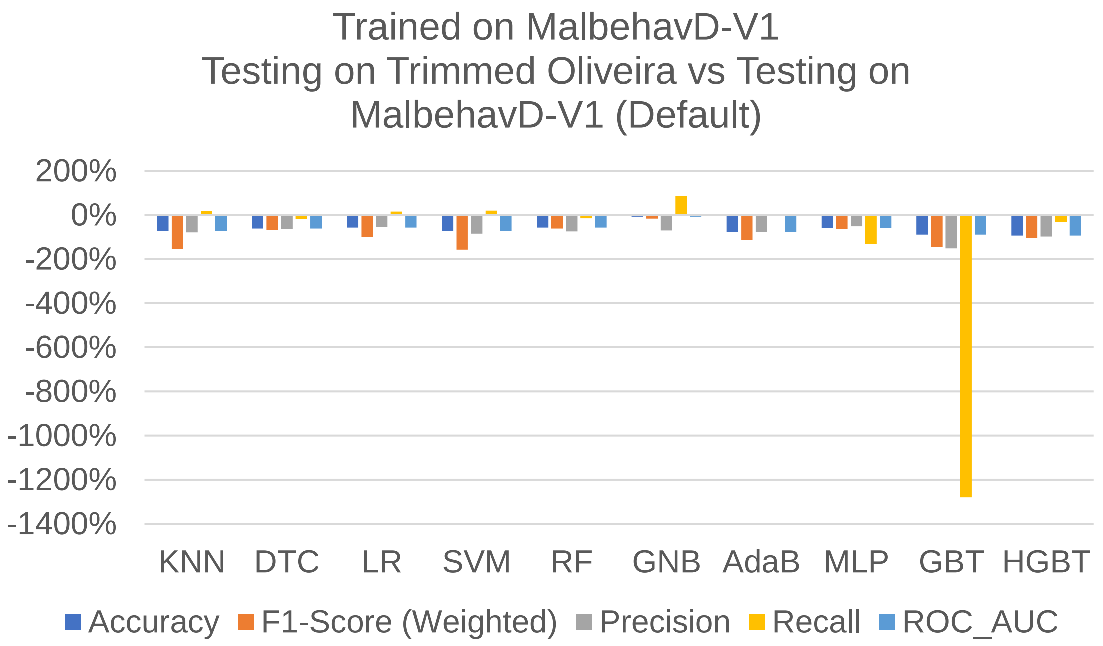
Model Robustness Test

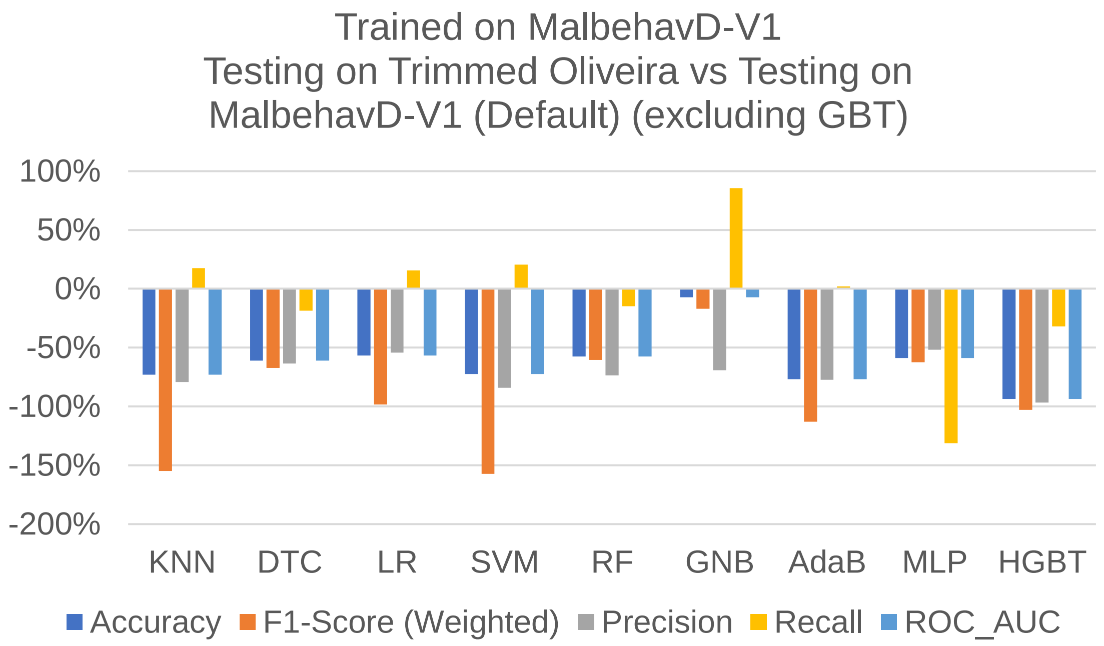




Results

Model Robustness Test

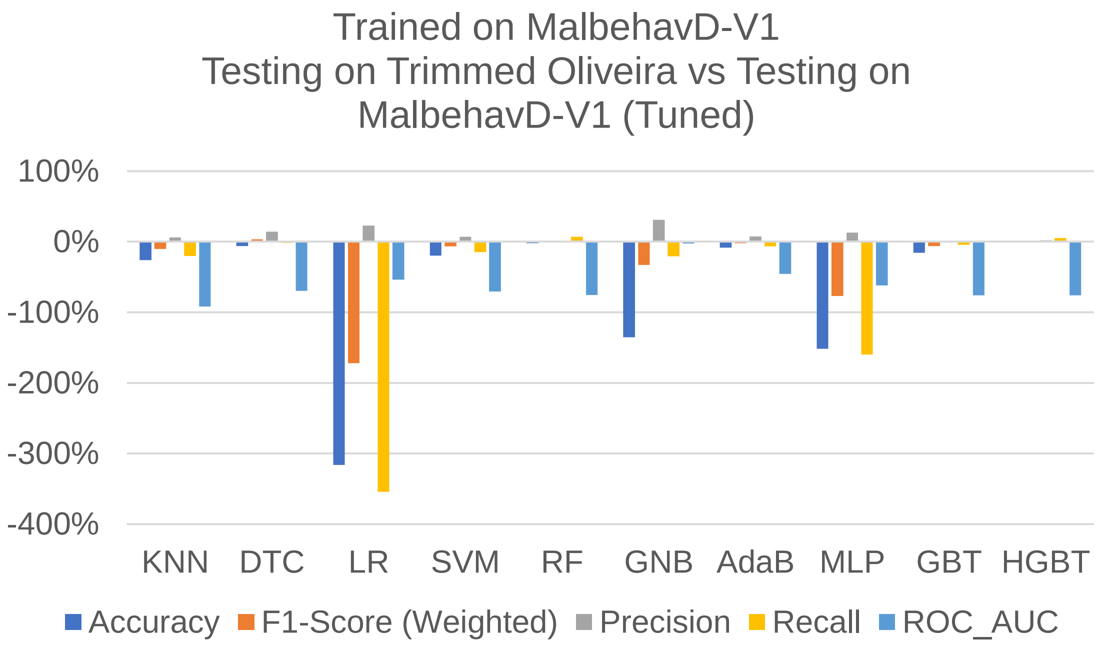


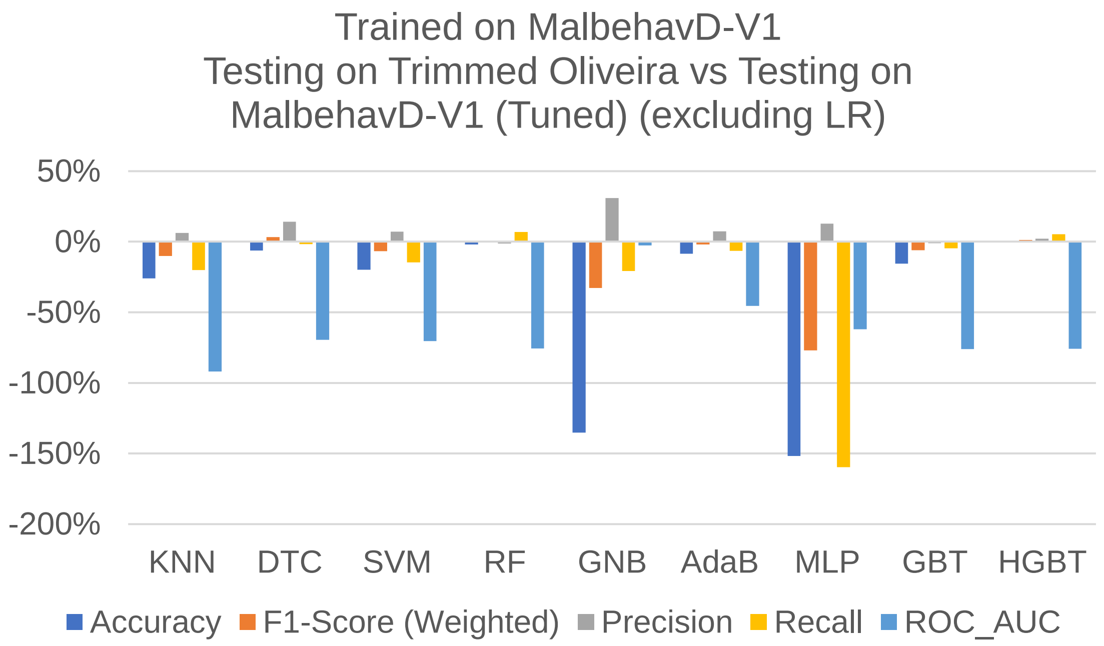


Testing on Trimmed Oliveira [+] vs Testing on MalbehavD-V1 [-]
(Default)

Results

Model Robustness Test

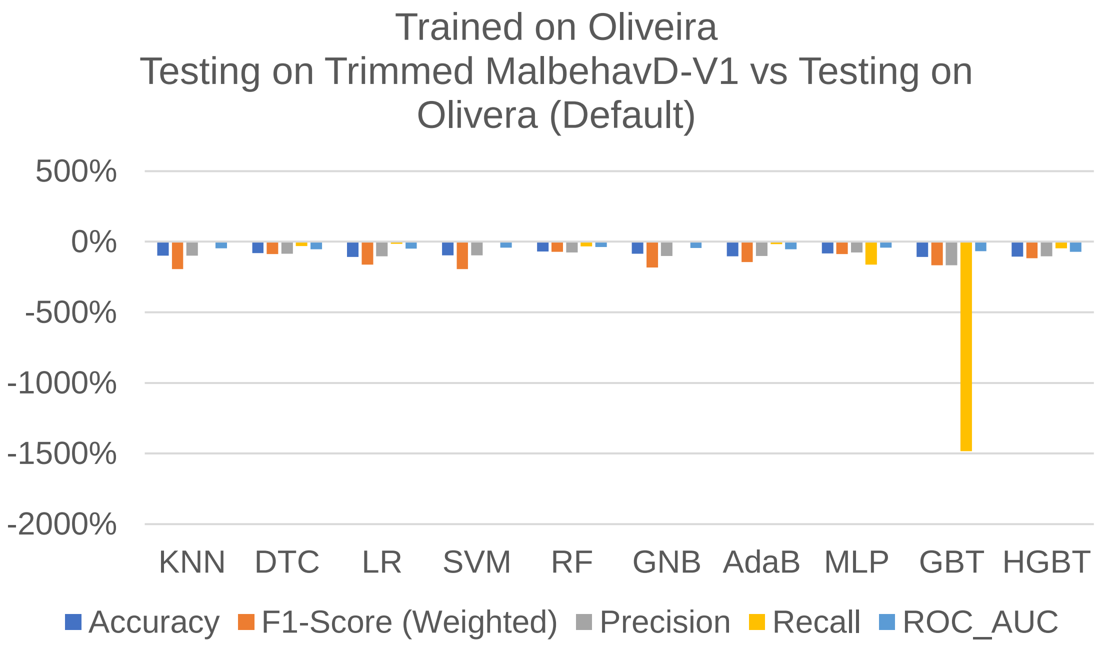


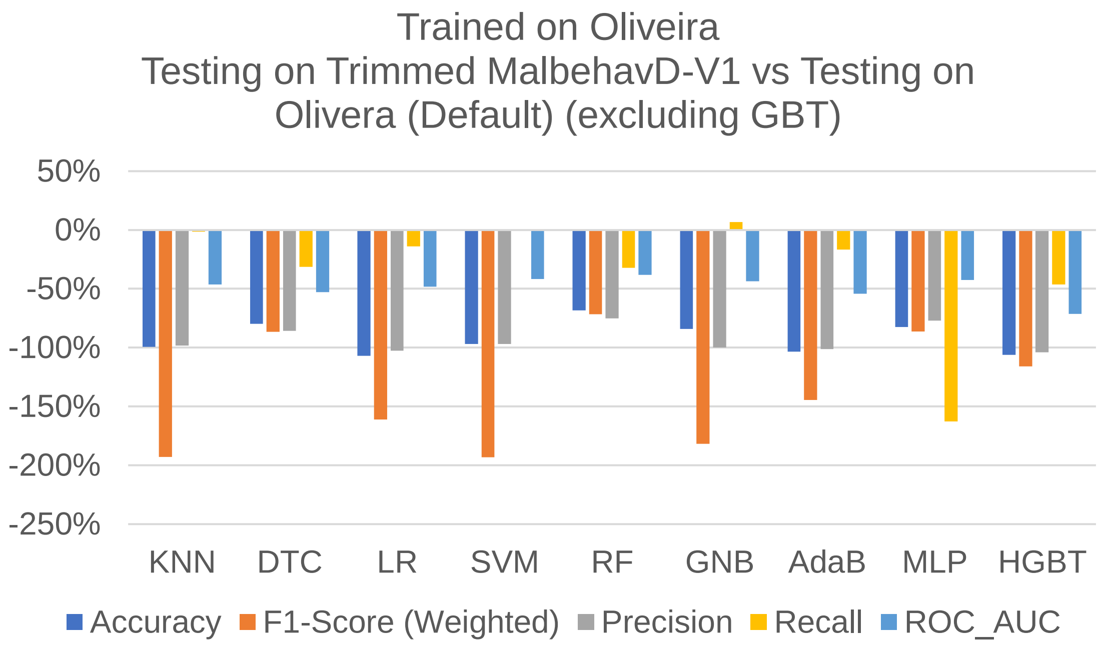


Testing on Trimmed Oliveira [+] vs Testing on MalbehavD-V1 [-] (Tuned)

Results

Model Robustness Test

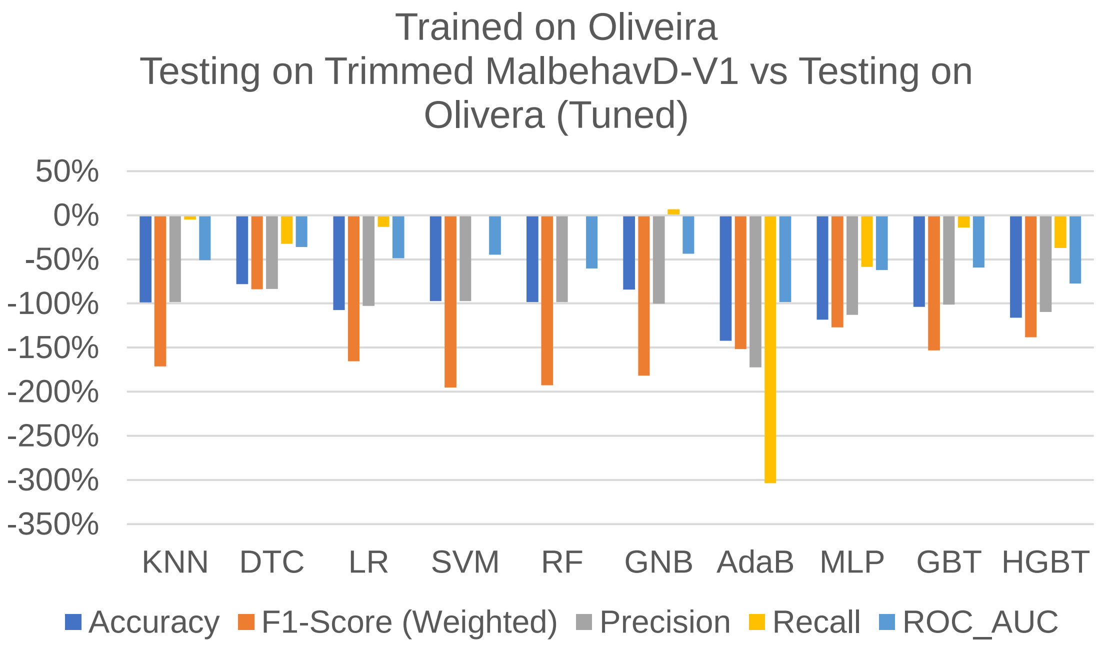


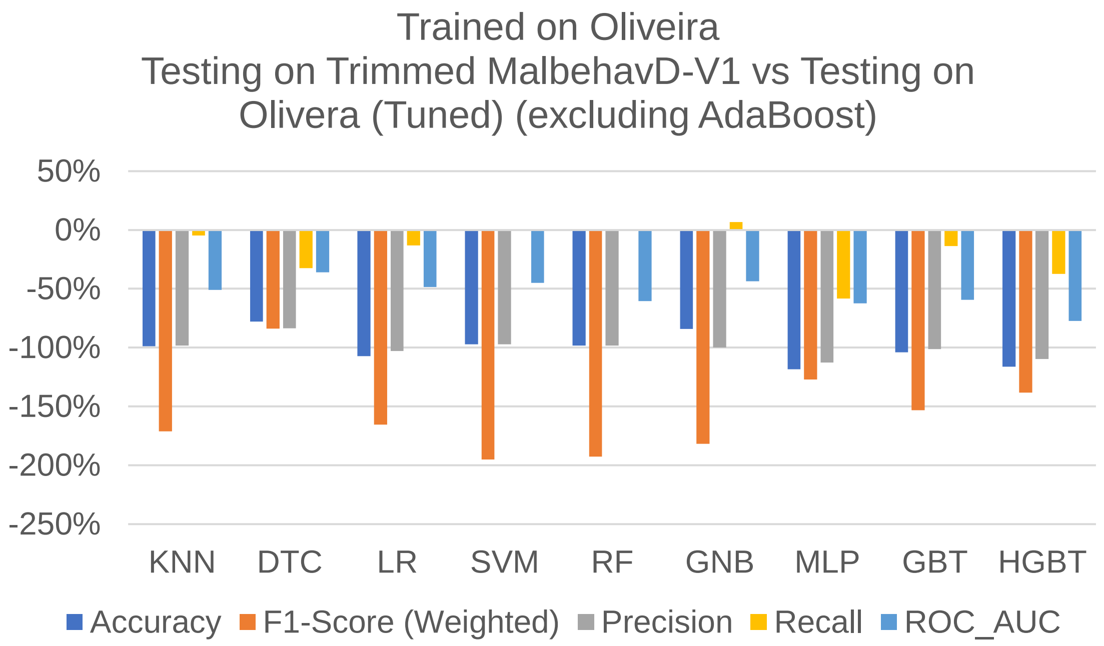


Testing on Trimmed MalbehavD-V1 [+] vs Testing on Oliveira [-] (Default)

Results

Model Robustness Test

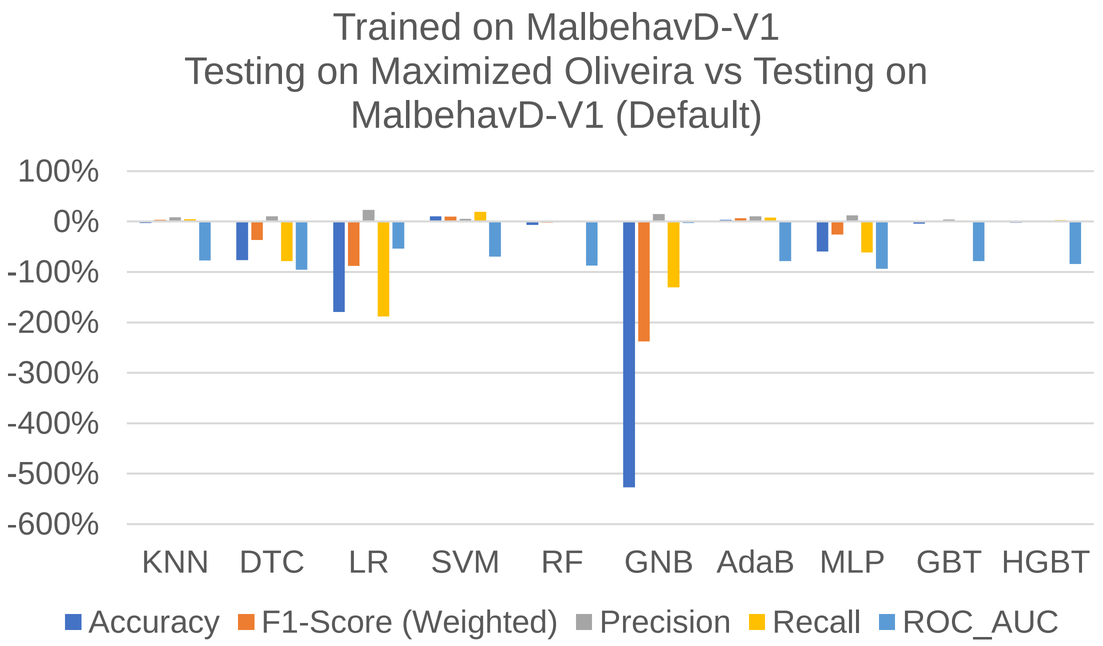


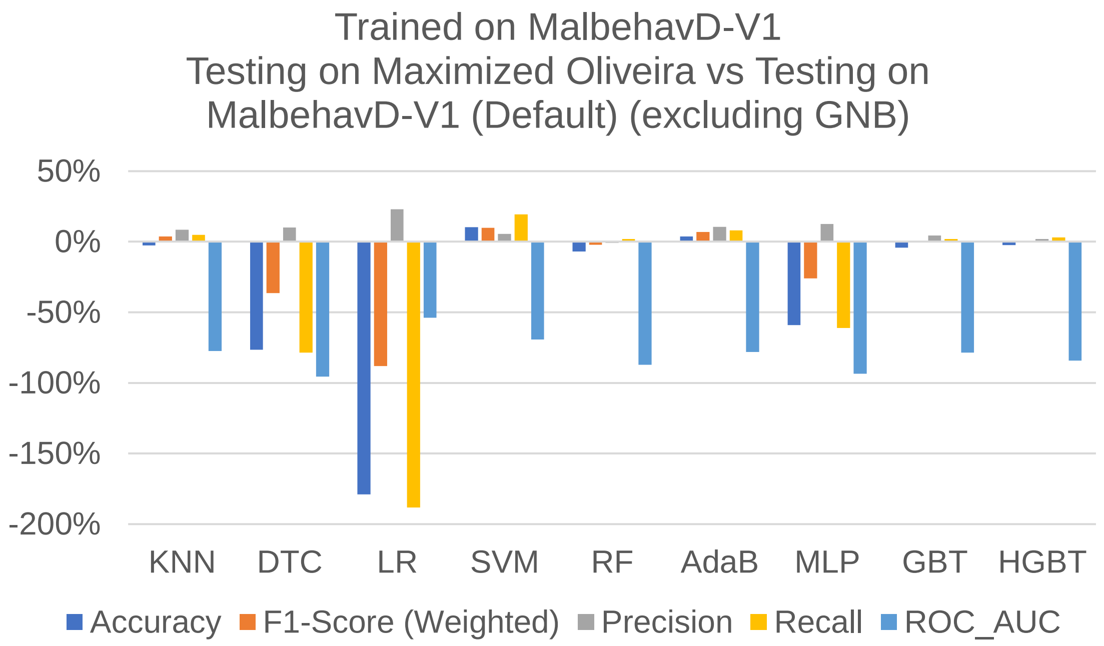


Testing on Trimmed MalbehavD-V1 [+] vs Testing on Oliveira [-]
(Tuned)

Results

Model Robustness Test

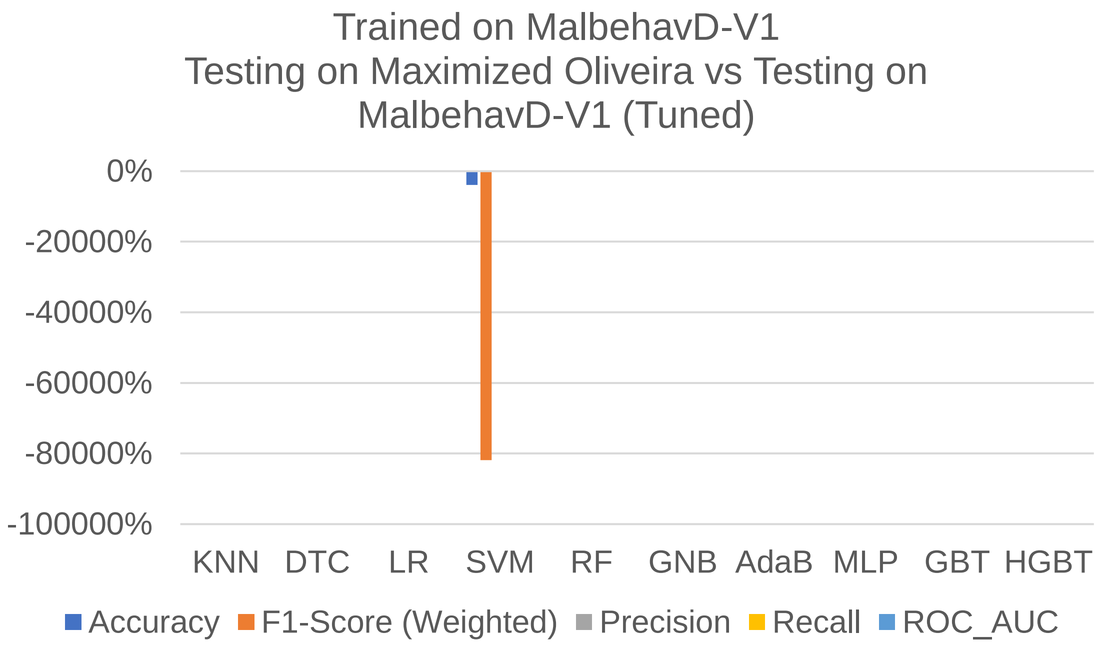


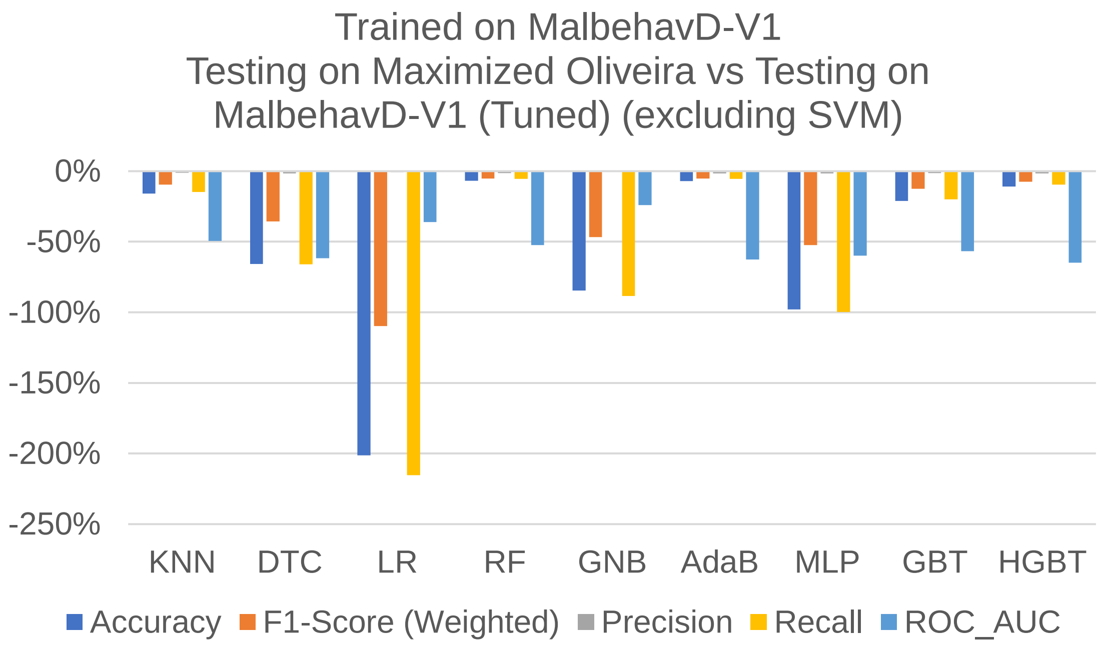


Testing on Maximized Oliveira [+] vs Testing on MalbehavD-V1 [-]
(Default)

Results

Model Robustness Test

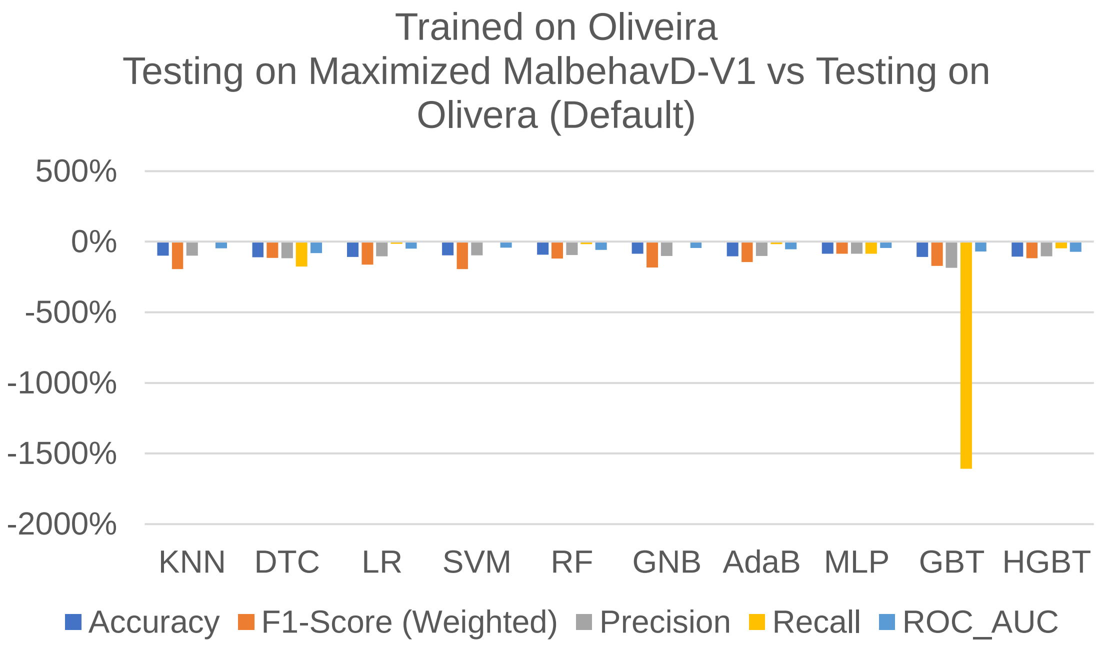


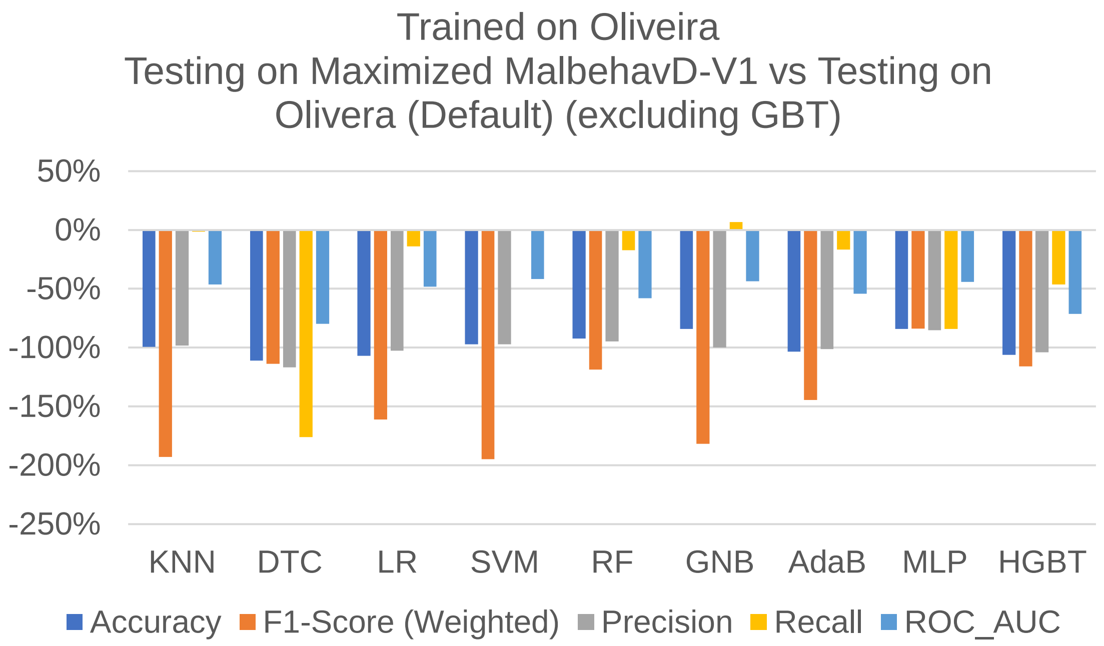


Testing on Maximized Oliveira [+] vs Testing on MalbehavD-V1 [-] (Tuned)

Results

Model Robustness Test

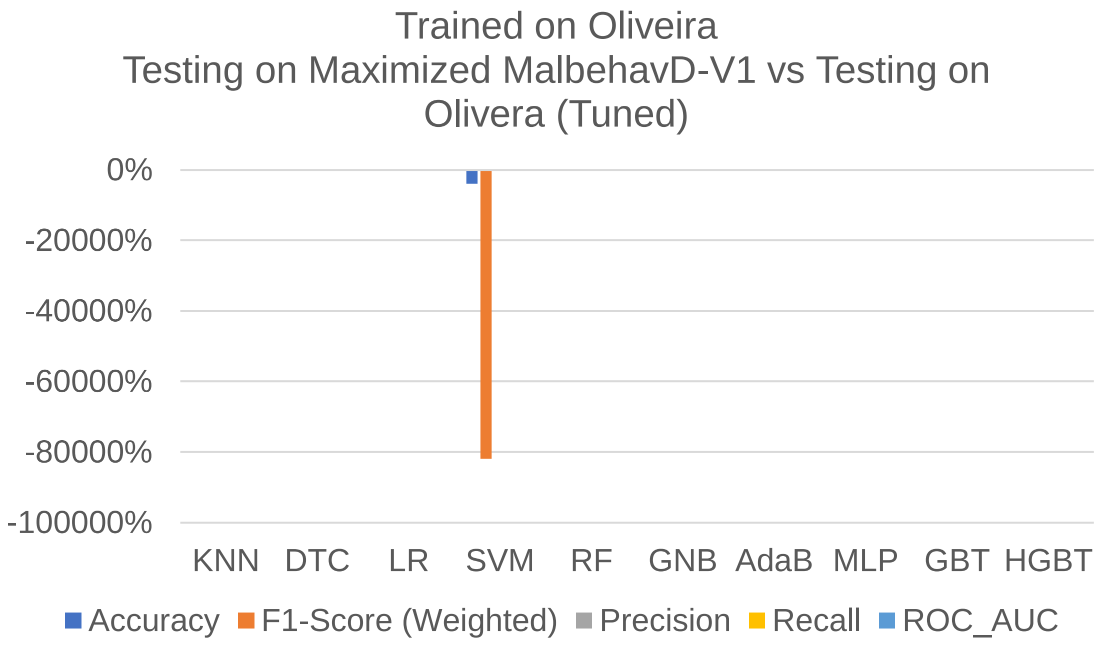


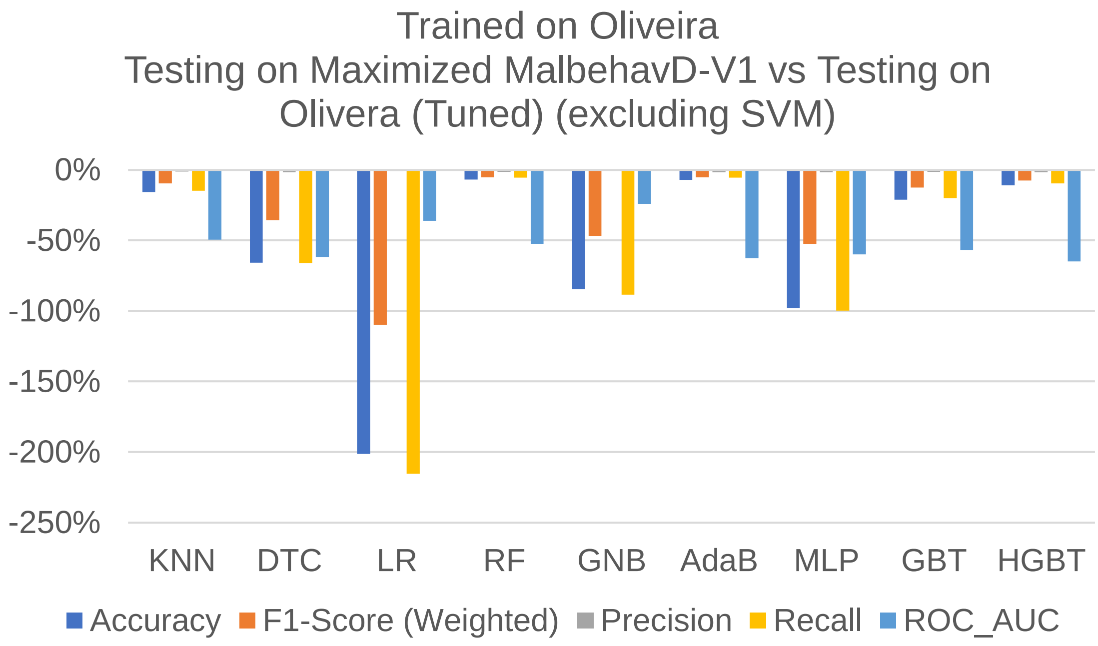


Testing on Maximized MalbehavD-V1 [+] vs Testing on Oliveira [-] (Default)

Results

Model Robustness Test

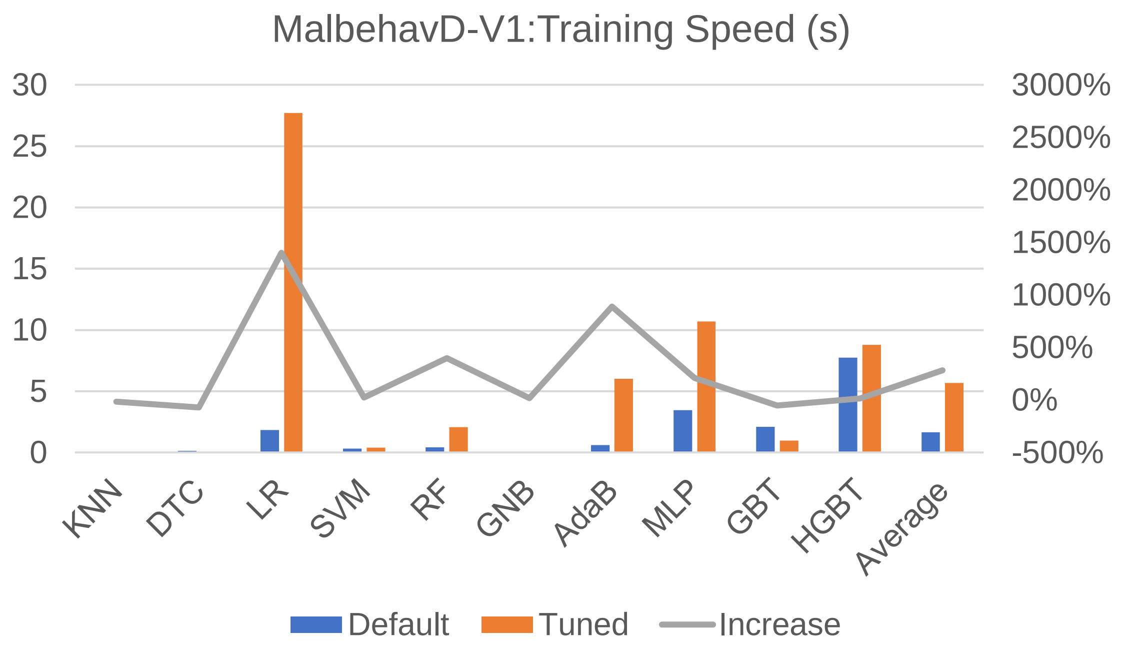


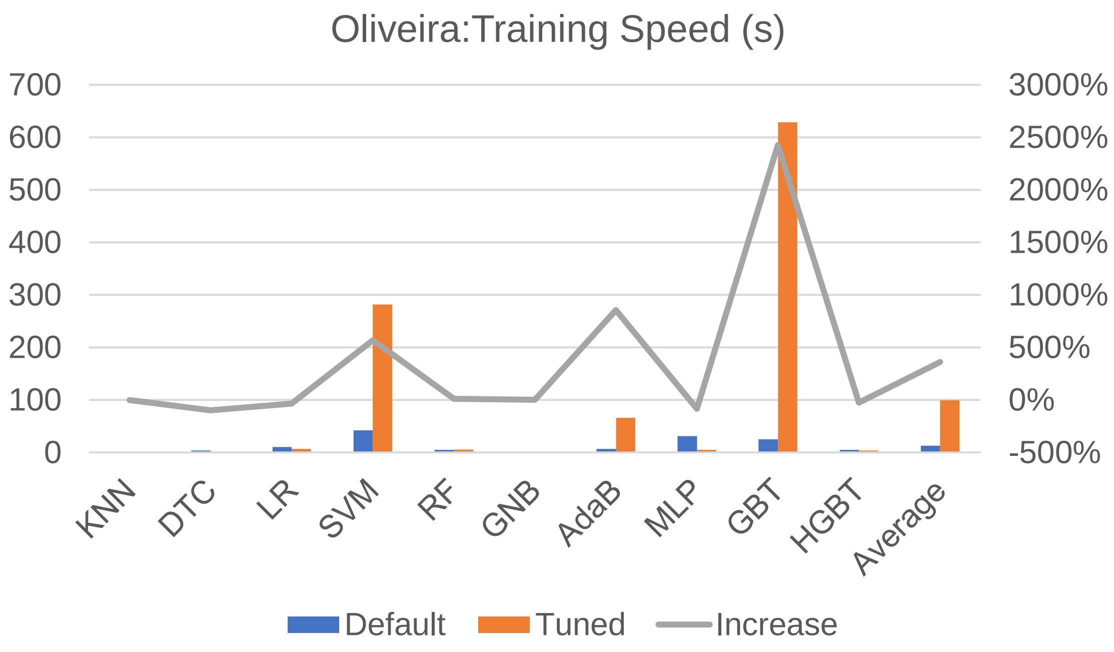


Testing on Maximized MalbehavD-V1 [+] vs Testing on Oliveira [-] (Tuned)

Results

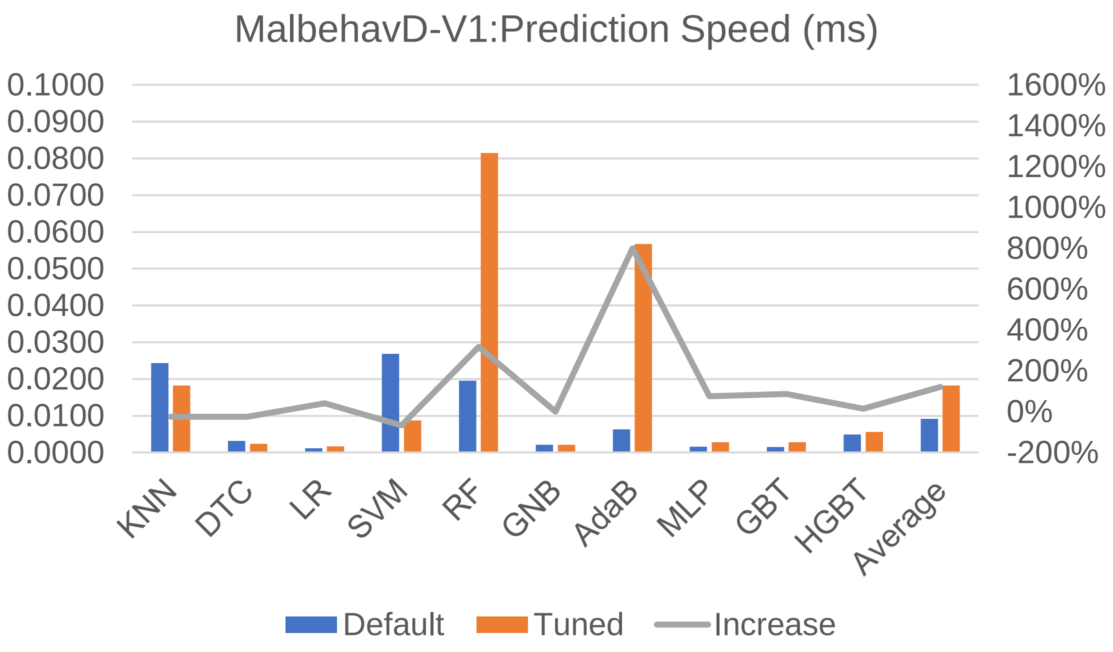
Time Test

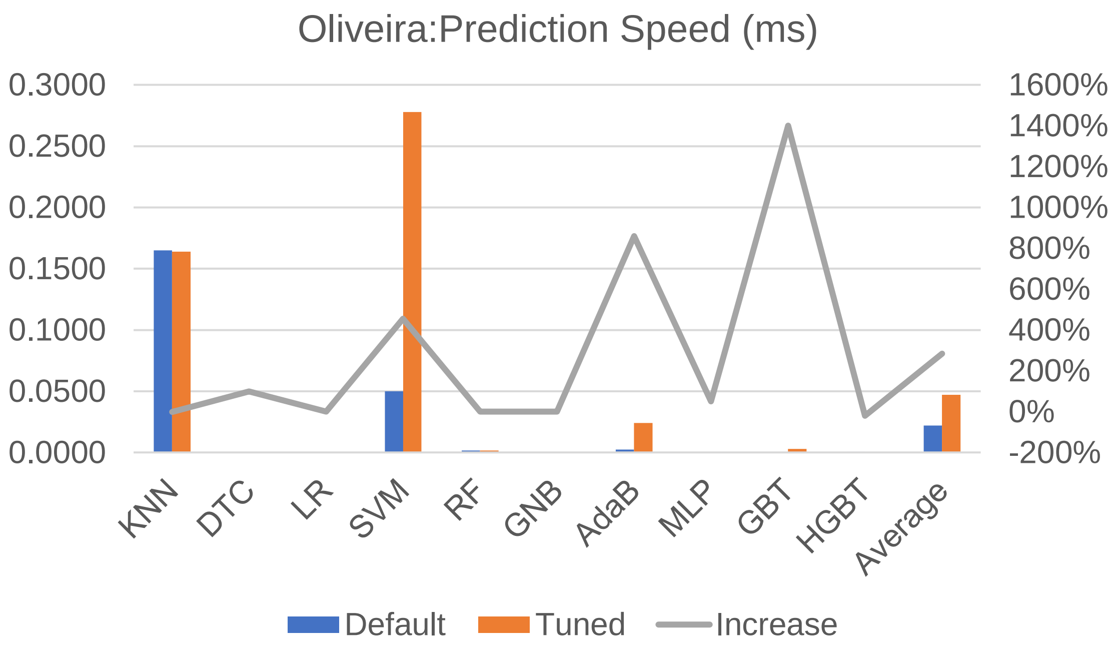




Results

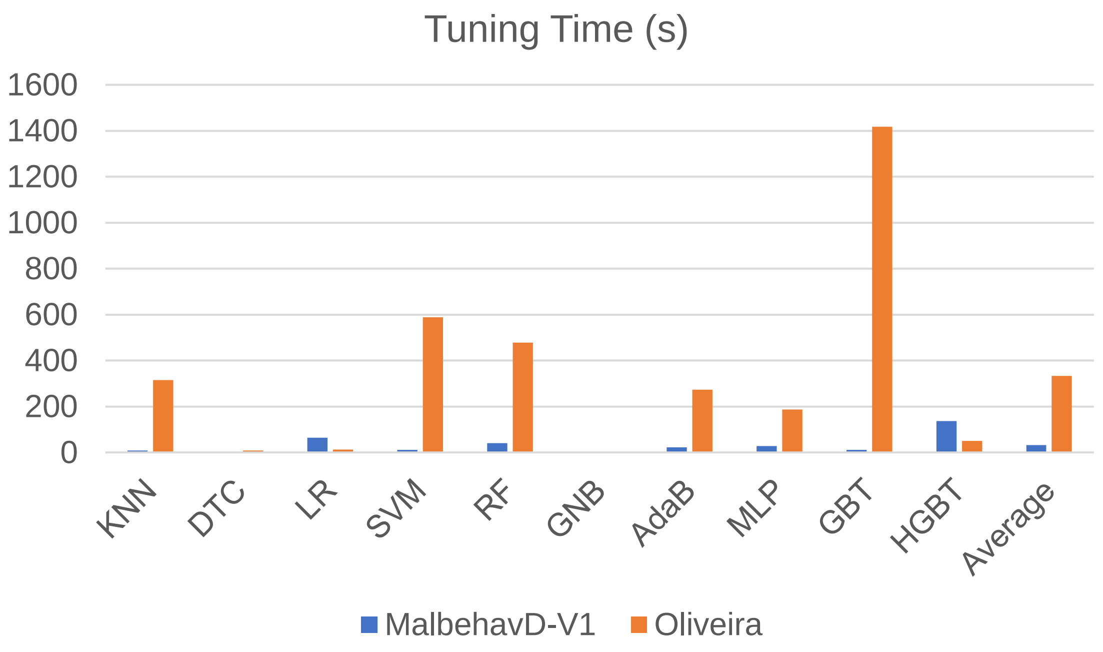
Time Test

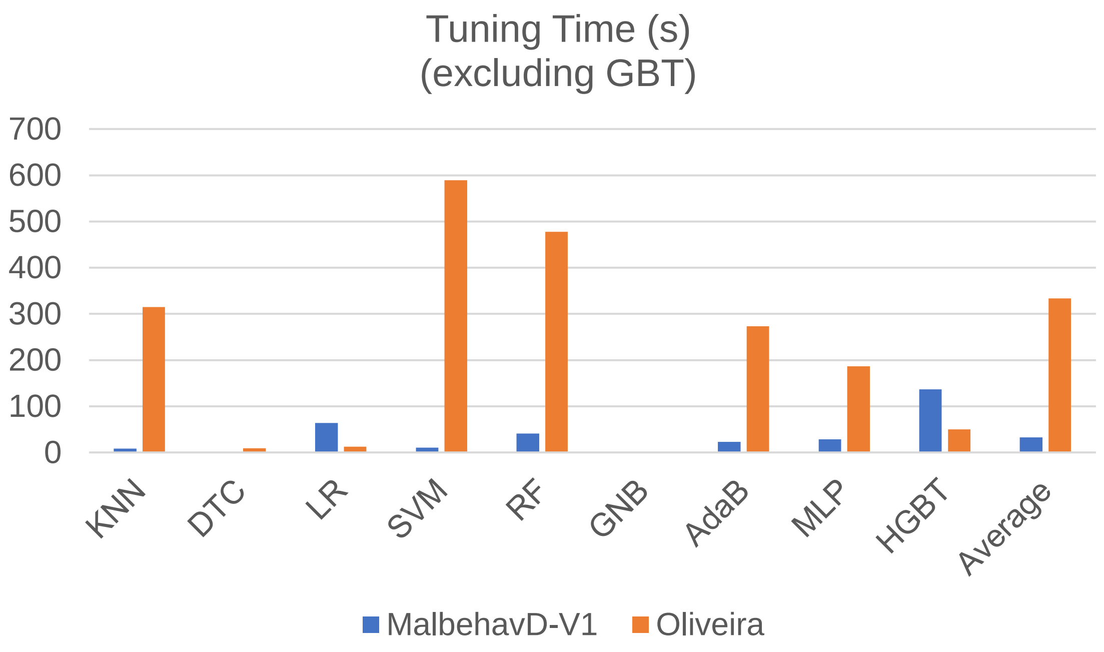




Results

Time Test





Observations, Interpretations, and Discussion of Data

Default vs Tuned Model Performance Comparison

- Most **models trained using MalbehavD-V1** (esp. traditional ones) **improved** in terms of performance after tuning.
- Most **models trained using Oliveira** did not incur any significant improvement in most metrics, except in ROC-AUC, after tuning. Only models such as AdaBoost and DTC experienced performance improvement and deterioration in ROC-AUC, respectively.
- Summary:
  - Tuning an ML model hyperparameters **can improve its performance** across a range of metrics, **if done correctly**.
  - The dataset used in training can also play a role in determining possible improvements, if any, after tuning as some datasets may benefit from tuning while others might not.
  - Ensemble models also had good results, mostly experiencing no changes after tuning implying an already near-optimal performance from its default tuning already.

Observations, Interpretations, and Discussion of Data

Dataset Performance Comparison

Metric	MalbehavD-V1	Oliveira
Accuracy	0.5210 (GNB) to 0.9300 (HGBT)	0.8948 (GNB) to 0.9901 (HGBT)
F1-Score (Weighted)	0.3839 (GNB) to 0.9299 (HGBT)	0.9252 (GNB) to 0.9892 (HGBT)
Precision	0.7498 (LR) to 0.9826 (HGBT)	0.9854 (GNB) to 0.9920 (HGBT)
Recall	0.1412 (GNB) to 0.9012 (HGBT)	0.9050 (GNB) to 0.9999 (SVM)
ROC-AUC	0.5214 (GNB) to 0.9300 (HGBT)	0.6981 (GNB) to 0.8382 (DTC)

Dataset Performance Range Values (Default)

Recall	MalbehavD-V1	Oliveira
0	0.0522 (GNB) to 0.9960 (RF)	0.3467 (LR) to 0.6436 (HGBT)
1	0.7019 (LR) to 0.9698 (RF)	0.9149 (GNB) to 0.9999 (SVM)

Dataset Performance in Individual Recall Values (Default)

Observations, Interpretations, and Discussion of Data

Dataset Performance Comparison

Metric	MalbehavD-V1	Oliveira
Accuracy	0.5514 (GNB) to 0.9288 (HGBT)	0.8948 (GNB) to 0.9895 (RF)
F1-Score (Weighted)	0.4652 (GNB) to 0.9287 (HGBT)	0.9252 (GNB) to 0.9884 (HGBT)
Precision	0.6796 (GNB) to 0.9885 (RF)	0.9854 (LR) to 0.9906 (HGBT)
Recall	0.2460 (GNB) to 0.9020 (AdaBoost)	0.9050 (GNB) to 0.9999 (SVM)
ROC-AUC	0.5516 (GNB) to 0.9288 (HGBT)	0.6981 (GNB) to 0.8120 (HGBT)

Dataset Performance Range Values (Tuned)

Recall	MalbehavD-V1	Oliveira
0	0.0522 (GNB) to 0.9960 (RF)	0.3378 (LR) to 0.6356 (AdaBoost)
1	0.7019 (LR) to 0.9698 (RF)	0.9149 (GNB) to 1.0000 (SVM)

Dataset Performance in Individual Recall Values (Tuned)

Observations, Interpretations, and Discussion of Data

Dataset Performance Comparison

- There is are some minor differences between the datasets of MalbehavD-V1 and Olivera where the latter performed best from overall performance perspective regardless if the model is tuned or not.
- The Oliveira dataset, however, is not perfect. Due to its great quantity imbalance between malicious and non-malicious samples, resulted to issues in the recall metric specifically for non-malicious samples.

Observations, Interpretations, and Discussion of Data

Model Robustness Test

- All models, regardless of trained dataset, does not result in high model robustness. However, interventions that improve the performance of the model when encountering unforeseen/untrained datasets.
- There is a minor to considerable increase in performance (in various metrics) when the tuned model was used instead of the default one which further suggests the advantages of hyperparameter tuning.
- There are mixed results in terms of reshaping technique used. MalbehavD-V1 performed well on all techniques while Oliveira only performed well on maximizing.
- Training on the Oliveira dataset has better model robustness which is due to its sample size despite its sample imbalance issue.

Observations, Interpretations, and Discussion of Data

Time Test

- Training and Tuning times are influenced by the complexity of the models and the size of the input dataset the model is trying to train from.
- Training time a tuned model, regardless of its innate complexity, is more than a non-tuned one.
- Prediction time is affected by the complexity of the trained model, especially if it is also tuned.
- System hardware capabilities also plays a role in the different time values.

Observations, Interpretations, and Discussion of Data

Other Observations

- There are certain models consume large amounts of memory which can easily consume available memory resources during processes like tuning where multiple instances of the model are being trained.
- There are certain models that are single-threaded which suggests that the single-core performance of the CPU might be important as much as multi-core performance is.
- Having high-speed storage may also be helpful as it may contribute to faster data read and write especially during dataset pre-processing and cleaning.

Conclusion

- There are various steps in building an ML model which involve the computing environment, dataset, ML model, and evaluation.
- Constant variables are as important as non-constant variables whenever ML models are being compared.
- Between MalbehavD-V1 and Oliveira, Oliveira had an overall better performance the prior. However, it has issues on individual recall values for benign samples and poor model robustness.

Conclusion

- Traditional models were mostly found to be relatively light to tune, train, and evaluate albeit at the expense of its performance across a broad range of metrics.
- Ensemble models were found to be quite heavy to tune, train, and evaluate, however at the advantage of having better performance across a broad range of metrics.

References

[1] M. Sewak, S. K. Sahay, and H. Rathore, “Comparison of Deep Learning and the Classical Machine Learning Algorithm for the Malware Detection,” in 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Busan: IEEE, Jun. 2018, pp. 293–296. doi: 10.1109/SNPD.2018.8441123.

[2] O. Aslan and R. Samet, “A Comprehensive Review on Malware Detection Approaches,” IEEE Access, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[3] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, “API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques,” J. Netw. Comput. Appl., vol. 218, p. 103704, Sep. 2023, doi: 10.1016/j.jnca.2023.103704.

[4] A. Oliveira, “Malware Analysis Datasets: API Call Sequences.” IEEE DataPort, Oct. 23, 2019. doi: 10.21227/TQQM-AQ14.

[5] F. O. Catak and A. F. Yazı, “A Benchmark API Call Dataset for Windows PE Malware Classification,” 2019, doi: 10.48550/ARXIV.1905.01999.

[6] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” J. Mach. Learn. Res., vol. 12, no. 85, pp. 2825–2830, 2011.

[7] B. J. Erickson and F. Kitamura, “Magician’s Corner: 9. Performance Metrics for Machine Learning Models,” Radiol. Artif. Intell., vol. 3, no. 3, p. e200126, May 2021, doi: 10.1148/ryai.2021200126.

[8] D. Chicco and G. Jurman, “The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification,” BioData Min., vol. 16, no. 1, p. 4, Feb. 2023, doi: 10.1186/s13040-023-00322-4.

[9] S. Prusty, S. Patnaik, and S. K. Dash, “SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer,” Front. Nanotechnol., vol. 4, 2022, Accessed: Aug. 17, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnano.2022.972421