POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Data and Information Quality Project 28: DQ Issue Dependency and ML Task Classification

Author(s): **Tommaso Aiello**

Group Number: **10687571**

Academic Year: 2023-2024

Contents

1 | Chapter One

1.1. Introduction

In this project for the Data and Information Quality course, I am looking into a specific problem in data quality called 'Feature Dependency'. This problem can affect the performance of Machine Learning models. The main goal is to understand how this issue changes the performance of these algorithms. To do this, I am conducting 10 different experiments. Each experiment uses a different scenario. This way, it is possible to see how feature dependency impacts the results in various situations.

The classification models used for this task are **Decision Tree**, **Logistic Regression**, **KNN**, **Random Forest**, **AdaBoost** and **MLP**. The classification metrics are the *F1 weighted score*, *the distance between train and test performances* (F1 weighted score) and *the speed of training* (time to train).

1.2. Setup Choices

1.2.1. Data collection phase

The base dataset is created using the sklearn function called *'make_ dataset_for_ classification'* and it is created using 5 features, 5 of which are informative, and with samples belonging to only two classes.

Starting from this base dataset some changes are introduced for each of the 10 experiments that were performed. The changes may involve introducing new features with a different correlation value or type. I have chosen to create some ad hoc utility functions to inject various type of correlated features instead of using the parameters of the *'make_ dataset_for_ classification'*. Those functions are present in the utility function section of the *'project.ipynb'* file.

1.2.2. Experiments

1. Introduce one correlated feature with an increasing correlation strength that starts from 0 to 1.
2. Introduce one correlated feature with a decreasing correlation strength that starts with 0 to -1.
3. Introduce a growing number of features with low correlation.
4. Introduce a growing number of features with medium correlation.
5. Introduce a growing number of features with a high correlation.
6. Introduce a growing number of features with a negative correlation.
7. Inject more and more non linear correlated features (power of 2 of a feature).
8. Inject more and more non linear correlated features (cosine).
9. Introducing new features as a linear combination of existing features.
10. Introducing 5 correlated features going from -1 to 1 correlation strength.

2 | Chapter Two

2.1. Pipeline Implementation

2.1.1. Data Collection

The base dataset is created using the *'make_ dataset_for_ classification'* function from sklearn library that was provided in the file *'A_ data_ collection.py'*. The parameters used are the default ones. So it creates a dataset composed of two classes with 5 informative features and 0 informative features.

2.1.2. Data Pollution Function

Experiment 1

Given the base dataset 10 new datasets are created. Each new dataset is composed of 6 features in which one feature has been injected. In each new dataset the new injected feature is going to have a correlation strength that is increasing starting from 0 to 1.

Experiment 2

Given the base dataset 10 new datasets are created. Each new dataset is composed of 6 features in which one feature has been injected. In each new dataset the new injected feature is going to have a correlation strength that is decreasing starting from 0 to -1.

Experiment 3

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be correlated with the original 5 features. The correlation strength is **low** and is around 0.40. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 4

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be correlated with the original 5 features. The correlation strength is **medium** and is around 0.65. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 5

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be correlated with the original 5 features. The correlation strength is **high** and is around 0.90. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 6

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be correlated with the original 5 features. The correlation strength is **negative** and is around -0.70. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 7

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be **non linearly** correlated with the original 5 features. The injected features are going to be **the power**

of 2 of the original features plus a noise. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 8

For the purpose of this experiment, 15 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 15 new datasets, 5 new features will be introduced so the number of features will grow more and more. These additional features are specifically designed to be **non linearly** correlated with the original 5 features. The injected features are going to be **the cosine of the original features** plus a noise. The shape of the base dataset is going to be (1000,5) while the shape of the 15th dataset is going to be (1000,80) so 75 are going to be injected features.

Experiment 9

For the purpose of this experiment, 10 new datasets will be created. Each of these datasets will be derived from the original dataset but with an added layer of complexity. In each of the 10 new datasets, a new feature will be introduced so the number of features will grow. This additional feature is specifically designed to be a **linear combination** of two of the original 5 features. The shape of the base dataset is going to be (1000,5) while the shape of the 10th dataset is going to be (1000,15).

Experiment 10

For the purpose of this experiment, 20 new datasets will be created. Each of these datasets will be derived from the original dataset. In each of the 20 new datasets, 5 new features will be introduced. This additional features are specifically designed to be have an increasing correlation strength starting from -1 to 1. The shape of all the datasets in this experiment is going to be (1000,10) and those 5 new added features are going to have a fixed correlation strength with respect to the 5 original features.

2.1.3. Data Analysis and evaluation

Each dataset has been evaluated using the models that were given in the *'D_ data_ analysis.py'*. The classification models used for this task are Decision Tree, Logistic Regression, KNN, Random Forest, AdaBoost and MLP.

In *'E_ plot_ results.py'* there are the given functions that were used to create the visualization of the results. For each experiment three plots are created showing the F1 weighted score, the distance between train and test performances (F1 weighted score) and the speed of training. Those plots can be used to compare the performances of different models on the different datasets that were created during the data pollution phase.

The code for these steps were provided and that code was not modified.

3 | Chapter Three

3.0.1. Results experiment 1

Introduce one correlated feature with an increasing correlation strength that starts from 0 to 1.

In Figure 3.1 it is shown the distance between f1 train and f1 test. Since only one correlated feature is injected there is not a visible difference in performance. The changes are really small and could be due to fluctuations also in the way the data is split for the train-val-test sets. Generally it is possible to observe that one of the values with the smaller distance (less overfitting) for every model is the one coming from the base dataset.

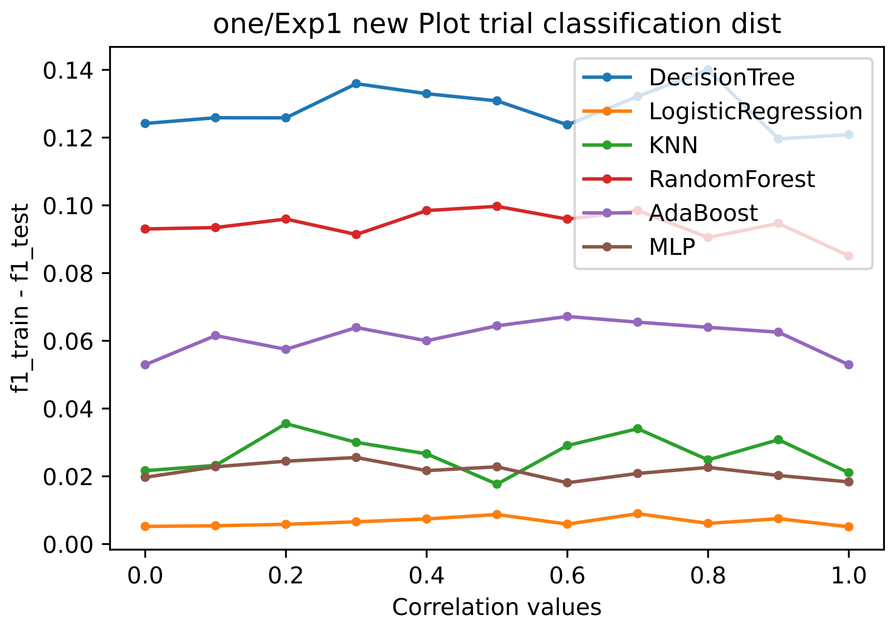

Figure 3.1: Exp1 distance between train and test performances

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1242	0.0052	0.0217	0.0930	0.0529	0.0197
0.1	0.1259	0.0054	0.0232	0.0934	0.0616	0.0228
0.2	0.1258	0.0058	0.0356	0.0960	0.0575	0.0245
0.3	0.1359	0.0066	0.0300	0.0914	0.0639	0.0256
0.4	0.1330	0.0074	0.0266	0.0985	0.0600	0.0217
0.5	0.1308	0.0087	0.0177	0.0997	0.0644	0.0228
0.6	0.1238	0.0059	0.0291	0.0959	0.0672	0.0181
0.7	0.1322	0.0090	0.0341	0.0985	0.0655	0.0208
0.8	0.1400	0.0061	0.0248	0.0905	0.0640	0.0226
0.9	0.1196	0.0075	0.0308	0.0947	0.0626	0.0202
1	0.1209	0.0051	0.0211	0.0851	0.0529	0.0183

Table 3.1: Exp 1 Distance

In Figure 3.2 it is shown the F1 weighted score for each model and each dataset. Generally there are not significant drops or benefits in performance in adding just one correlated feature and the oscillation of the result more than due to the correlation value may be due to the way data is split into train-val-test sets.

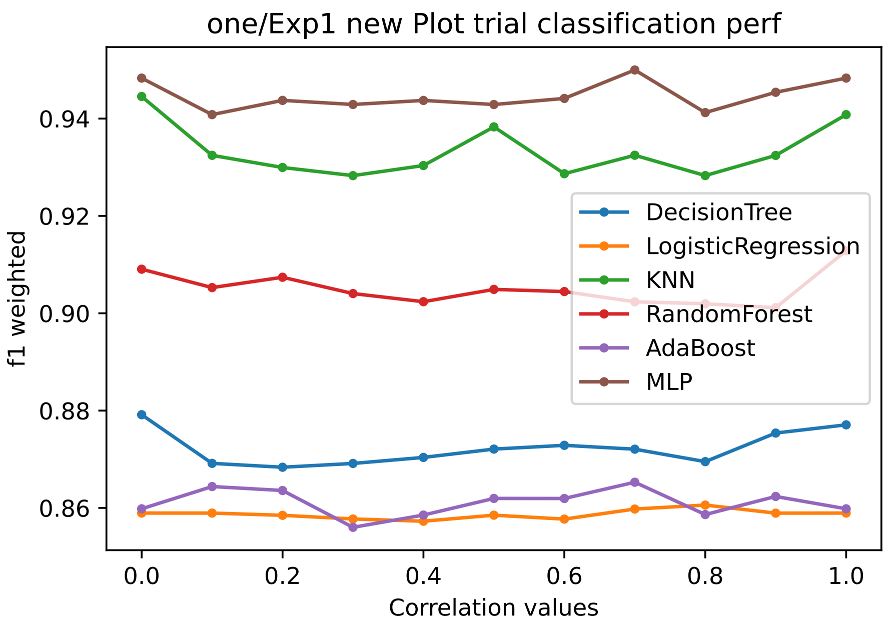

Figure 3.2: Exp1 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8821	0.8589	0.9445	0.9103	0.8598	0.9454
5	0.8654	0.8544	0.8814	0.8977	0.8553	0.9287
10	0.8475	0.8527	0.8693	0.8801	0.8531	0.9074
15	0.8467	0.8494	0.8530	0.8793	0.8544	0.8957
20	0.8334	0.8460	0.8356	0.8730	0.8498	0.8807
25	0.8321	0.8482	0.8281	0.8688	0.8486	0.8795
30	0.8282	0.8448	0.8348	0.8706	0.8507	0.8745
35	0.8283	0.8511	0.8386	0.8697	0.8473	0.8719
40	0.8233	0.8482	0.8361	0.8672	0.8377	0.8691
45	0.8258	0.8416	0.8441	0.8743	0.8381	0.8724
50	0.8245	0.8416	0.8365	0.8672	0.8399	0.8687
55	0.8191	0.8371	0.8533	0.8638	0.8341	0.8716
60	0.8140	0.8379	0.8515	0.8617	0.8366	0.8633
65	0.8198	0.8287	0.8507	0.8673	0.8411	0.8621
70	0.8082	0.8295	0.8553	0.8623	0.8441	0.8629
75	0.8107	0.8399	0.8533	0.8661	0.8424	0.8682

Table 3.2: Exp1 Mean Performance

In Figure 3.3 it is shown the speed of training that remains almost constant for every model because in this experiment only one more feature is introduced so it does not affect the time performance by a lot. It is possible to see a little increment in the training of the **MLP** which is the most time consuming model among the ones that were given.

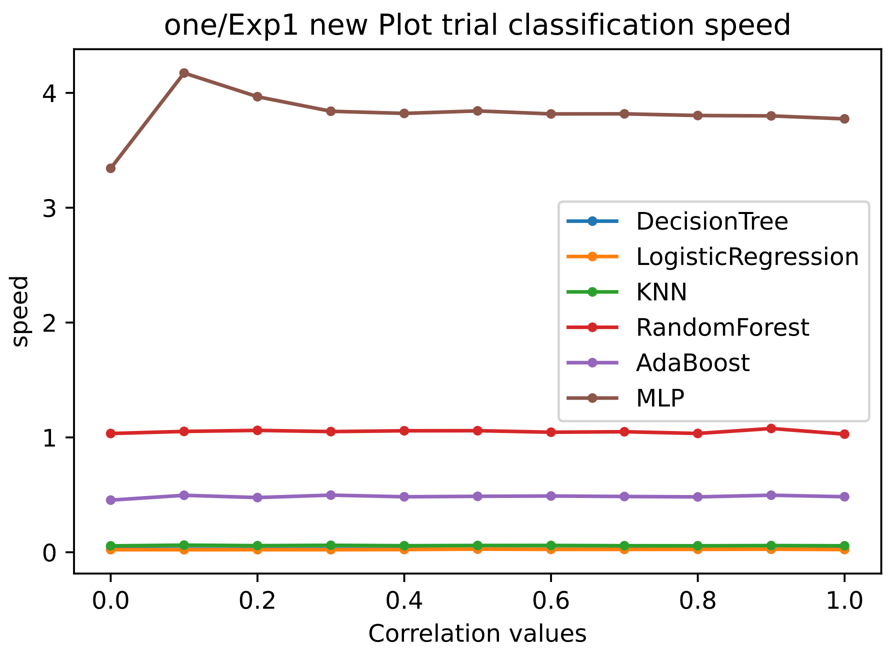

Figure 3.3: Exp1 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0324	0.0220	0.0747	1.1386	0.4556	3.7147
5	0.0421	0.0279	0.0799	1.2262	0.5760	4.2539
10	0.0622	0.0348	0.0767	1.3249	0.7081	4.7663
15	0.0859	0.0376	0.0774	1.4801	0.8200	5.0639
20	0.1099	0.0373	0.0773	1.6487	0.9384	5.1150
25	0.1310	0.0413	0.0905	1.6459	1.0615	5.4559
30	0.1527	0.0504	0.0931	1.7100	1.1796	6.0217
35	0.1675	0.0531	0.0920	1.8528	1.3059	5.5069
40	0.1848	0.0647	0.0924	1.8599	1.4279	5.3625
45	0.2141	0.0741	0.0917	2.0414	1.5466	6.0234
50	0.2676	0.0805	0.0950	2.0459	1.6674	5.5706
55	0.2581	0.0703	0.0949	2.0515	1.7947	5.8045
60	0.2861	0.0835	0.0950	2.2326	1.9119	5.4096
65	0.3139	0.0790	0.0957	2.2567	2.0304	5.2626
70	0.3307	0.0834	0.0982	2.2414	2.1448	5.3028
75	0.3524	0.0887	0.0983	2.2519	2.2937	5.3842

Table 3.3: Exp 1 Speed

3.0.2. Results experiment 2

Introduce one correlated feature with a decreasing correlation strength that starts with 0 to -1.

In Figure 3.4 it is shown the distance between the f1 train and f1 test which is a metric that can be used to investigate if the model is overfitting. The explanation for this experiment is very similar to the one of the Experiment one. Having a negative correlated injected feature is not changing the scenario a lot with respect to injecting a feature with positive correlation. Click here to see previous explanation

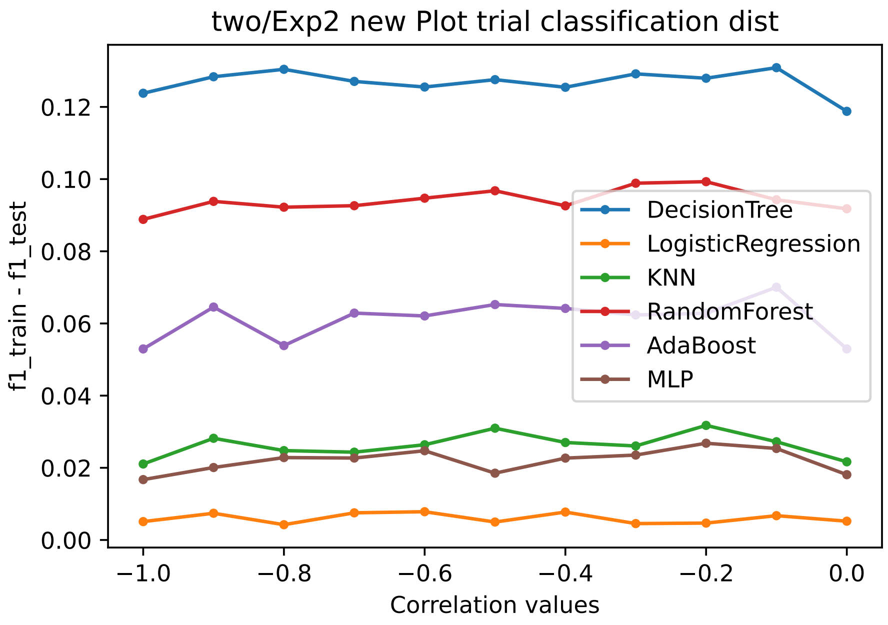

Figure 3.4: Exp2 distance between train and test performances

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1188	0.0052	0.0217	0.0918	0.0529	0.0181
-0.1	0.1309	0.0067	0.0272	0.0943	0.0700	0.0254
-0.2	0.1279	0.0047	0.0318	0.0993	0.0628	0.0268
-0.3	0.1292	0.0045	0.0261	0.0989	0.0624	0.0235
-0.4	0.1254	0.0077	0.0270	0.0926	0.0642	0.0227
-0.5	0.1276	0.0050	0.0310	0.0968	0.0652	0.0185
-0.6	0.1255	0.0078	0.0264	0.0947	0.0621	0.0247
-0.7	0.1271	0.0075	0.0243	0.0926	0.0629	0.0227
-0.8	0.1304	0.0042	0.0248	0.0922	0.0539	0.0228
-0.9	0.1284	0.0074	0.0282	0.0938	0.0645	0.0201
-1	0.1238	0.0051	0.0211	0.0888	0.0529	0.0167

Table 3.4: Exp 2 Distance

In Figure 3.5 it is shown the f1 weighted score for each model on each dataset. It is possible to see that there are not significant drops in performance as the correlation values changes (as it can be seen for the positive values). But the results of the models of the base dataset are one of the best results showing that having a new correlated feature, if not informative, is not beneficial but in most of the other cases is even producing a negative effect.

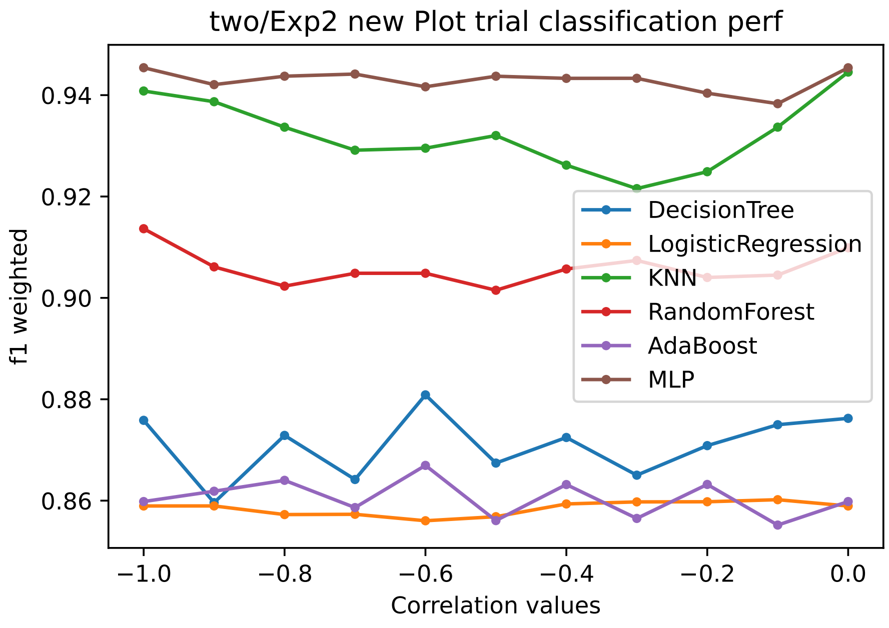

Figure 3.5: Exp2 F1 weighted score

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8762	0.8589	0.9445	0.9099	0.8598	0.9454
-0.1	0.8750	0.8602	0.9337	0.9045	0.8552	0.9383
-0.2	0.8709	0.8598	0.9249	0.9040	0.8632	0.9404
-0.3	0.8650	0.8598	0.9216	0.9074	0.8565	0.9433
-0.4	0.8725	0.8593	0.9262	0.9057	0.8632	0.9433
-0.5	0.8674	0.8568	0.9320	0.9015	0.8561	0.9437
-0.6	0.8809	0.8560	0.9295	0.9049	0.8670	0.9416
-0.7	0.8642	0.8573	0.9291	0.9049	0.8586	0.9442
-0.8	0.8729	0.8573	0.9337	0.9023	0.8640	0.9437
-0.9	0.8596	0.8590	0.9387	0.9061	0.8619	0.9421
-1	0.8758	0.8589	0.9408	0.9136	0.8598	0.9454

Table 3.5: Exp 2 Mean Performance

In Figure 3.6 it is shown the speed of training that remains almost constant for every model because in this experiment only one more feature is introduced so it does not affect the time performance by a lot. It is possible to see a little increment in the training of the **MLP** which is the most time consuming model among the ones that were given.

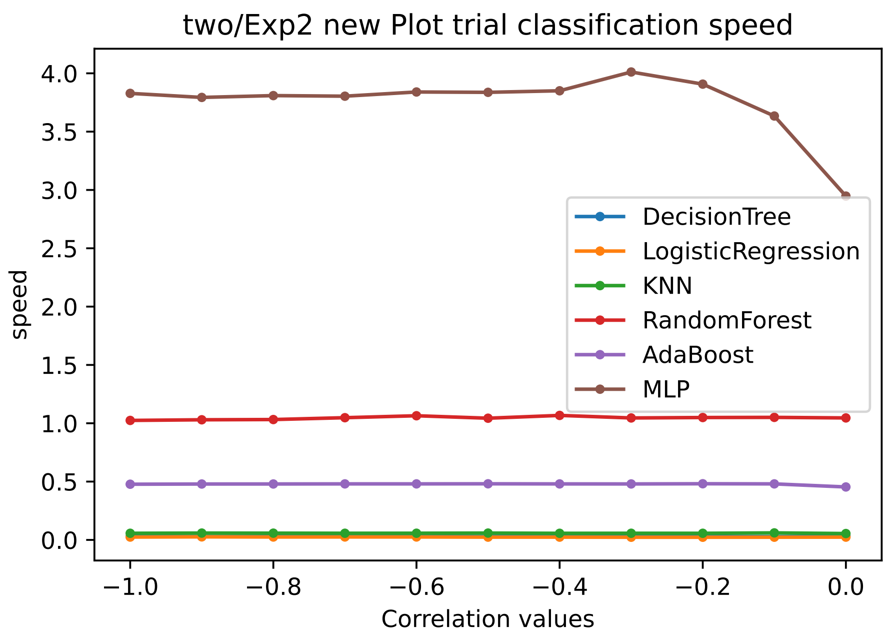

Figure 3.6: Exp2 speed of training

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0333	0.0239	0.0547	1.0458	0.4540	2.9475
-0.1	0.0279	0.0233	0.0602	1.0505	0.4804	3.6338
-0.2	0.0279	0.0229	0.0569	1.0492	0.4815	3.9071
-0.3	0.0291	0.0230	0.0570	1.0456	0.4798	4.0112
-0.4	0.0437	0.0239	0.0567	1.0673	0.4801	3.8502
-0.5	0.0280	0.0239	0.0587	1.0433	0.4811	3.8369
-0.6	0.0284	0.0252	0.0574	1.0645	0.4804	3.8399
-0.7	0.0302	0.0253	0.0571	1.0478	0.4804	3.8039
-0.8	0.0290	0.0250	0.0579	1.0319	0.4795	3.8089
-0.9	0.0289	0.0262	0.0587	1.0299	0.4792	3.7933
-1	0.0282	0.0245	0.0569	1.0244	0.4775	3.8280

Table 3.6: Exp 2 Speed

3.0.3. Results experiment 3

Introduce a growing number of features with low correlation.

In Figure 3.7 it is shown the distance between f1 train and f1 test. It is possible to observe that as the number of non informative feature grows the model tends to overfit because the distance between those value increases.

Injecting a big number of correlated features while keeping fixed the number of samples can cause the so called effect 'Curse of Dimensionality'. The **curse of dimensionality** and **overfitting** are interconnected because high-dimensional data often increases the risk of overfitting. When you have many features in your dataset, the complexity of the model can increase, making it more likely to fit the noise in the data rather than the true underlying relationships.

This is why it's crucial to carefully preprocess and manage high-dimensional data and use appropriate techniques to mitigate overfitting, such as regularization and feature selection or dimensionality reduction.

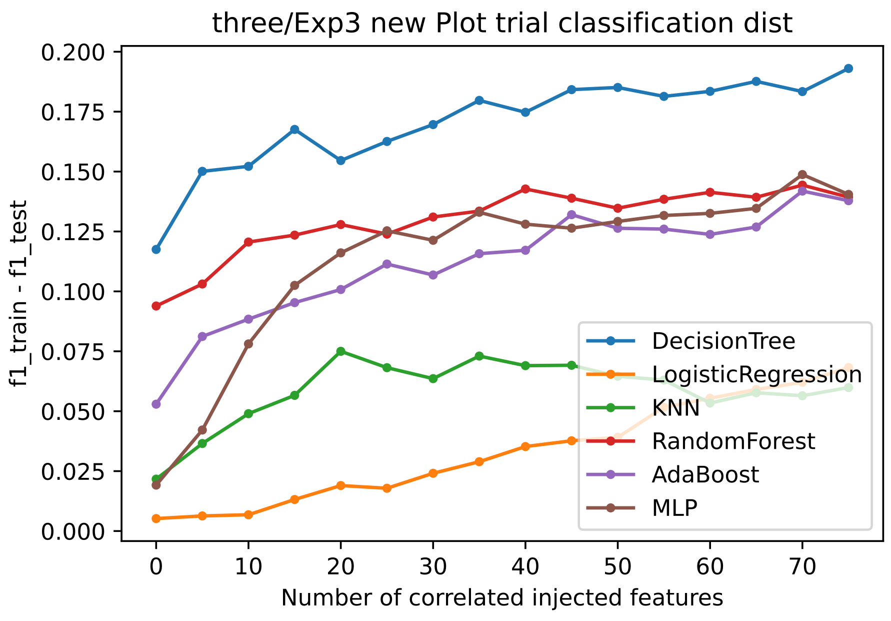

Figure 3.7: Exp3 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1175	0.0052	0.0217	0.0939	0.0529	0.0192
5	0.1501	0.0063	0.0365	0.1031	0.0812	0.0422
10	0.1522	0.0068	0.0490	0.1206	0.0884	0.0781
15	0.1676	0.0132	0.0566	0.1235	0.0953	0.1025
20	0.1546	0.0190	0.0750	0.1279	0.1008	0.1161
25	0.1626	0.0179	0.0682	0.1239	0.1114	0.1253
30	0.1696	0.0241	0.0636	0.1310	0.1068	0.1213
35	0.1797	0.0289	0.0730	0.1335	0.1157	0.1330
40	0.1747	0.0353	0.0690	0.1427	0.1172	0.1280
45	0.1842	0.0377	0.0692	0.1389	0.1320	0.1264
50	0.1851	0.0391	0.0646	0.1347	0.1263	0.1292
55	0.1814	0.0516	0.0630	0.1384	0.1260	0.1317
60	0.1835	0.0554	0.0534	0.1413	0.1238	0.1326
65	0.1876	0.0591	0.0577	0.1393	0.1269	0.1346
70	0.1834	0.0621	0.0565	0.1443	0.1419	0.1488
75	0.1930	0.0682	0.0599	0.1393	0.1378	0.1405

Table 3.7: Exp 3 Distance

In figure 3.8 it is shown the f1 weighted score for each model on each dataset. It is possible to see how the performance of all the models have a significant drop as new features with low correlation are injected.

Decision Tree: The performance decreases a lot at first but then it stabilizes. Decision trees can handle a large number of features without significant performance loss, but they can be sensitive to changes in the data.

Logistic Regression: There's a steady decline in performance, which suggests that adding more low-correlated features may introduce noise that hinders the model's ability to find a linear decision boundary. But there is not a big drop.

KNN: The F1 score drops sharply and then fluctuates. KNN is sensitive to feature scaling and irrelevant features, which can greatly impact performance. It is the model that has the most significant drop being the best model for the base dataset and the worst model when 25 correlated features are injected.

Random Forest: After an initial drop, the performance starts to stabilize. Random Forests are robust to irrelevant features due to feature bagging, but a large number of these features can still cause some loss of performance.

AdaBoost: It is one of the most stable models among the ones that were selected.

MLP (Multi-Layer Perceptron): The performance declines significantly as more features are added. MLPs can overfit to noise in the data, and with an increasing number of irrelevant features, the network might struggle to generalize.

Overall, the plot indicates that while all models are affected by the injection of low-correlated features, some are more resilient than others. AdaBoost that is an ensemble method shows a relative good stabilization. Also RandomForest that is an ensemble method shows that after an initial drop it has a stabilization in performance. A Simpler model like KNN shows a more pronounced decline in performance. Instead for MLP and DecisionTree it is possible to see a big drop at around 20 injected features and then they stabilizes.

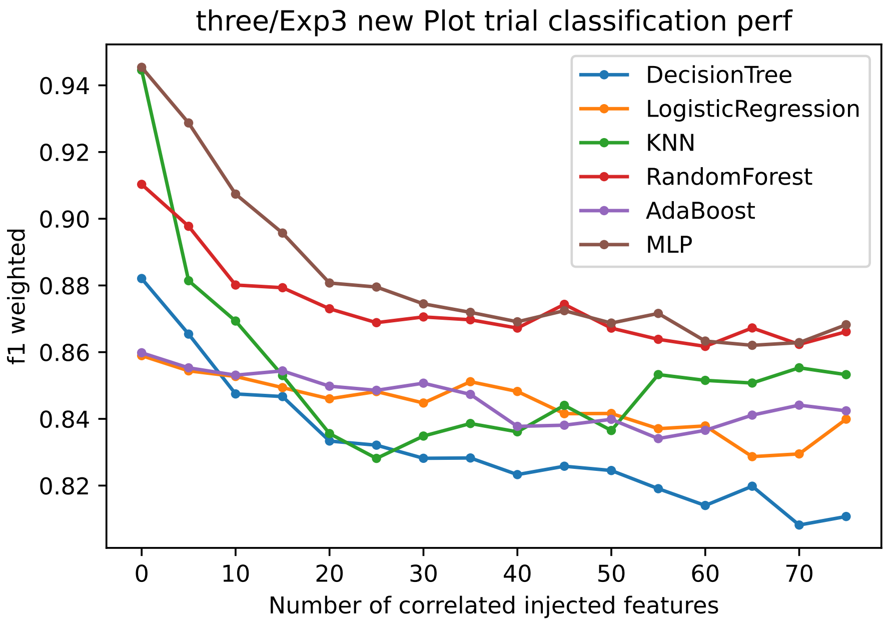

Figure 3.8: Exp3 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8821	0.8589	0.9445	0.9103	0.8598	0.9454
5	0.8654	0.8544	0.8814	0.8977	0.8553	0.9287
10	0.8475	0.8527	0.8693	0.8801	0.8531	0.9074
15	0.8467	0.8494	0.8530	0.8793	0.8544	0.8957
20	0.8334	0.8460	0.8356	0.8730	0.8498	0.8807
25	0.8321	0.8482	0.8281	0.8688	0.8486	0.8795
30	0.8282	0.8448	0.8348	0.8706	0.8507	0.8745
35	0.8283	0.8511	0.8386	0.8697	0.8473	0.8719
40	0.8233	0.8482	0.8361	0.8672	0.8377	0.8691
45	0.8258	0.8416	0.8441	0.8743	0.8381	0.8724
50	0.8245	0.8416	0.8365	0.8672	0.8399	0.8687
55	0.8191	0.8371	0.8533	0.8638	0.8341	0.8716
60	0.8140	0.8379	0.8515	0.8617	0.8366	0.8633
65	0.8198	0.8287	0.8507	0.8673	0.8411	0.8621
70	0.8082	0.8295	0.8553	0.8623	0.8441	0.8629
75	0.8107	0.8399	0.8533	0.8661	0.8424	0.8682

Table 3.8: Exp 3 Mean Performance

In Figure 3.9 (and also Figure 3.12, Figure 3.15, Figure 3.18, Figure 3.21, Figure 3.24) it is shown the time performance of the models for the experiment 3. Generally, the increase in training time as the number of features increases can be attributed to **computational complexity** (more features often mean more calculations), to **memory usage** (increased features can lead to higher memory consumption) and to **algorithm complexity** (some algorithms might need to evaluate interactions between features).

The linear increase in training time for **AdaBoost** and **Random Forest** algorithms as the number of features increases can be explained by the nature of these algorithms.

AdaBoost is an algorithm that builds a series of weak learners (typically decision trees) sequentially. Each new learner focuses on the mistakes of the previous ones. As the number of features increases, AdaBoost needs to process more information at each step, but since it works sequentially, the increase in processing time tends to be linear rather than exponential.

Random Forest is an algorithm that creates multiple decision trees, each trained on a random subset of features and data points. The increase in features means each tree has more potential splits to evaluate, but since trees are built independently, the overall increase in computation grows linearly with the number of features.

Even though **Decision Trees** are relatively efficient, they still need to evaluate each feature to determine the best splits at each node. As the number of features increases, the algorithm needs to perform more calculations to assess the best splitting criteria for each feature. This process, although not overly complex, does incur a slight increase in computational time as the feature set grows. However, since Decision Trees don't require complex transformations or interactions between features, the increase in training time is only marginal compared to more complex algorithms.

For **K-Nearest Neighbors (KNN)**, the training time remains almost constant regardless of the number of features because of the nature of the algorithm. KNN is a lazy learning algorithm, meaning it doesn't actually learn a model during the training phase. Instead, it stores the training data and performs computations only during the prediction phase. Therefore, the training phase is typically very fast and not significantly affected by the number of features in the dataset. The major computation for KNN, which involves finding the nearest neighbors, occurs during prediction, not during training.

For **Logistic Regression** even if the increase in time is not clearly visible in the plot due to the scale of the plot, it is possible to see an increase in the table relative to the speed, getting 4 times bigger compared to the base dataset example.

The significant increase in training time for a Multi-Layer Perceptron **(MLP)** with an increasing number of features is primarily due to its architecture and computational complexity. MLPs, being a type of neural network, consist of multiple layers of neurons with weights and biases that need to be adjusted during training. As the number of features increases, the input layer of the MLP expands, requiring more weights and biases to be tuned. This leads to a larger number of computations during the forward and backward passes of the training process. Additionally, the optimization process becomes more complex with more features, as the algorithm needs to find optimal values in a higher-dimensional space. This overall increase in complexity and computational burden results in a significant increase in training time for MLPs as the number of features grows.

These motivations explain the behaviour of the time performance of the models in all the others experiments that include the injection of a significant number of features. So these motivations are valid to explain also Figure 3.12, Figure 3.15, Figure 3.18, Figure 3.21, Figure 3.24.

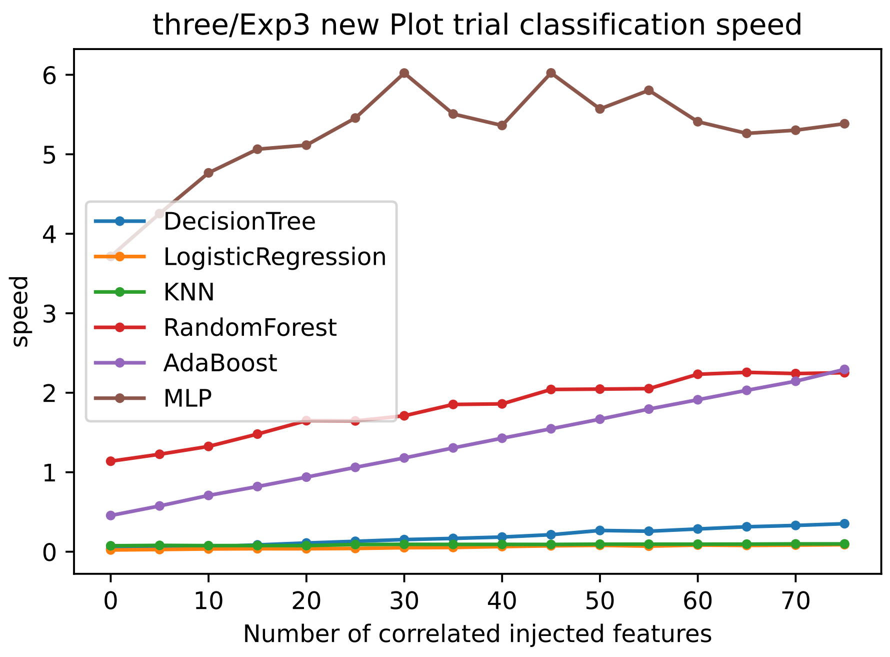

Figure 3.9: Exp3 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0324	0.0220	0.0747	1.1386	0.4556	3.7147
5	0.0421	0.0279	0.0799	1.2262	0.5760	4.2539
10	0.0622	0.0348	0.0767	1.3249	0.7081	4.7663
15	0.0859	0.0376	0.0774	1.4801	0.8200	5.0639
20	0.1099	0.0373	0.0773	1.6487	0.9384	5.1150
25	0.1310	0.0413	0.0905	1.6459	1.0615	5.4559
30	0.1527	0.0504	0.0931	1.7100	1.1796	6.0217
35	0.1675	0.0531	0.0920	1.8528	1.3059	5.5069
40	0.1848	0.0647	0.0924	1.8599	1.4279	5.3625
45	0.2141	0.0741	0.0917	2.0414	1.5466	6.0234
50	0.2676	0.0805	0.0950	2.0459	1.6674	5.5706
55	0.2581	0.0703	0.0949	2.0515	1.7947	5.8045
60	0.2861	0.0835	0.0950	2.2326	1.9119	5.4096
65	0.3139	0.0790	0.0957	2.2567	2.0304	5.2626
70	0.3307	0.0834	0.0982	2.2414	2.1448	5.3028
75	0.3524	0.0887	0.0983	2.2519	2.2937	5.3842

Table 3.9: Exp 3 Speed

3.0.4. Results experiment 4

Introduce a growing number of features with medium correlation.

In Figure 3.10 it is shown the distance between f1 train and f1 test. In general the performance are pretty similar to the ones of the experiment three. So overall the correlation value form low to medium did not have a significant impact on the overfitting for most of the models.

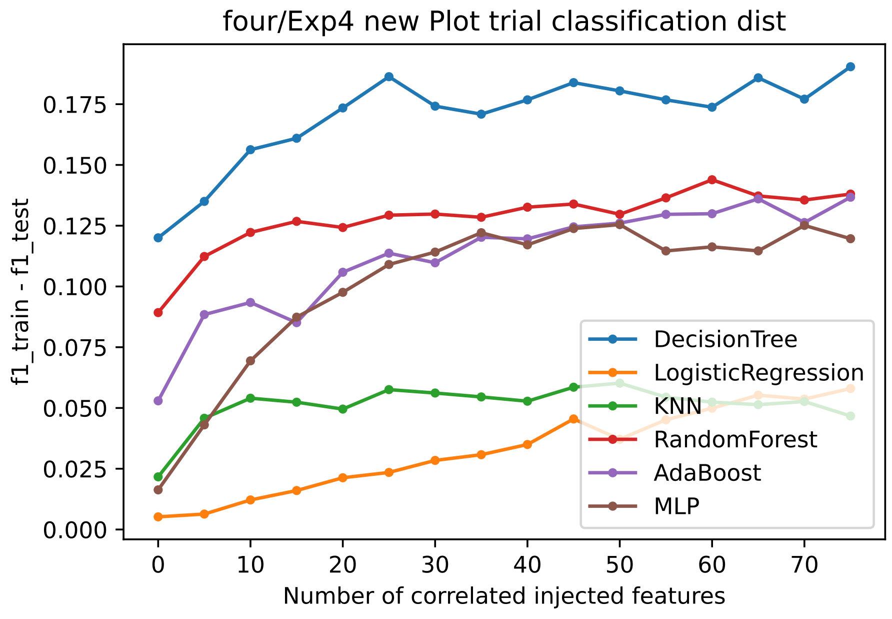

Figure 3.10: Exp4 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1200	0.0052	0.0217	0.0892	0.0529	0.0163
5	0.1350	0.0063	0.0458	0.1123	0.0884	0.0431
10	0.1563	0.0122	0.0540	0.1222	0.0934	0.0694
15	0.1610	0.0160	0.0524	0.1268	0.0851	0.0874
20	0.1734	0.0213	0.0495	0.1243	0.1058	0.0976
25	0.1863	0.0235	0.0576	0.1293	0.1137	0.1090
30	0.1742	0.0284	0.0562	0.1298	0.1098	0.1141
35	0.1709	0.0308	0.0545	0.1285	0.1202	0.1221
40	0.1768	0.0350	0.0528	0.1326	0.1196	0.1171
45	0.1839	0.0455	0.0585	0.1339	0.1245	0.1238
50	0.1805	0.0371	0.0602	0.1297	0.1261	0.1254
55	0.1768	0.0451	0.0545	0.1364	0.1297	0.1146
60	0.1737	0.0499	0.0524	0.1439	0.1299	0.1163
65	0.1858	0.0553	0.0514	0.1372	0.1360	0.1146
70	0.1771	0.0536	0.0527	0.1356	0.1263	0.1251
75	0.1904	0.0580	0.0467	0.1380	0.1367	0.1196

Table 3.10: Exp 4 Distance

In Figure 3.11 it is shown the f1 weighted score for each model on each dataset. As for experiment 3 **AdaBoost** is still the most stable model (even though there are more fluctuations). Random Forest has a similar behaviour as experiment 3, and overall Decision Tree too. Linear regression has a slightly more evident decrease trend as the number of injected features increases. MLP has a similar trend with respect to experiment 3 having a drop at around 20 injected features and stabilizing but the drop is smaller.

The model that is behaving differently is **KNN**. For KNN, the smaller drop in F1 score with medium-correlated features compared to low-correlated ones might be explained by its reliance on feature similarity to make predictions. Low-correlated features may act more like noise, reducing the algorithm's ability to accurately identify neighbors and make correct predictions.

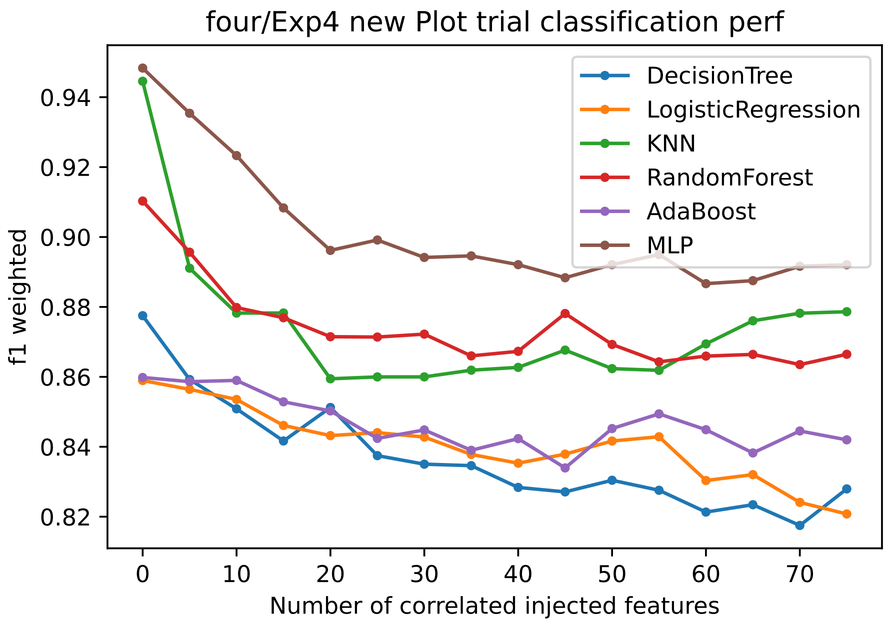

Figure 3.11: Exp4 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8775	0.8589	0.9445	0.9103	0.8598	0.9483
5	0.8592	0.8564	0.8911	0.8957	0.8586	0.9354
10	0.8509	0.8535	0.8782	0.8798	0.8590	0.9233
15	0.8417	0.8461	0.8783	0.8769	0.8529	0.9083
20	0.8512	0.8432	0.8594	0.8715	0.8503	0.8962
25	0.8375	0.8440	0.8600	0.8714	0.8424	0.8991
30	0.8350	0.8428	0.8600	0.8722	0.8448	0.8942
35	0.8346	0.8378	0.8619	0.8660	0.8390	0.8946
40	0.8284	0.8353	0.8627	0.8673	0.8423	0.8921
45	0.8271	0.8379	0.8676	0.8781	0.8339	0.8883
50	0.8304	0.8416	0.8624	0.8693	0.8452	0.8921
55	0.8276	0.8429	0.8619	0.8643	0.8494	0.8950
60	0.8213	0.8304	0.8694	0.8659	0.8449	0.8867
65	0.8234	0.8320	0.8760	0.8664	0.8382	0.8875
70	0.8175	0.8241	0.8782	0.8635	0.8445	0.8916
75	0.8279	0.8208	0.8786	0.8665	0.8420	0.8921

Table 3.11: Exp 4 Mean Performance

The explanation of the plot in Figure 3.12 is the same as the one for experiment 3. **Click here to go to the explanation**

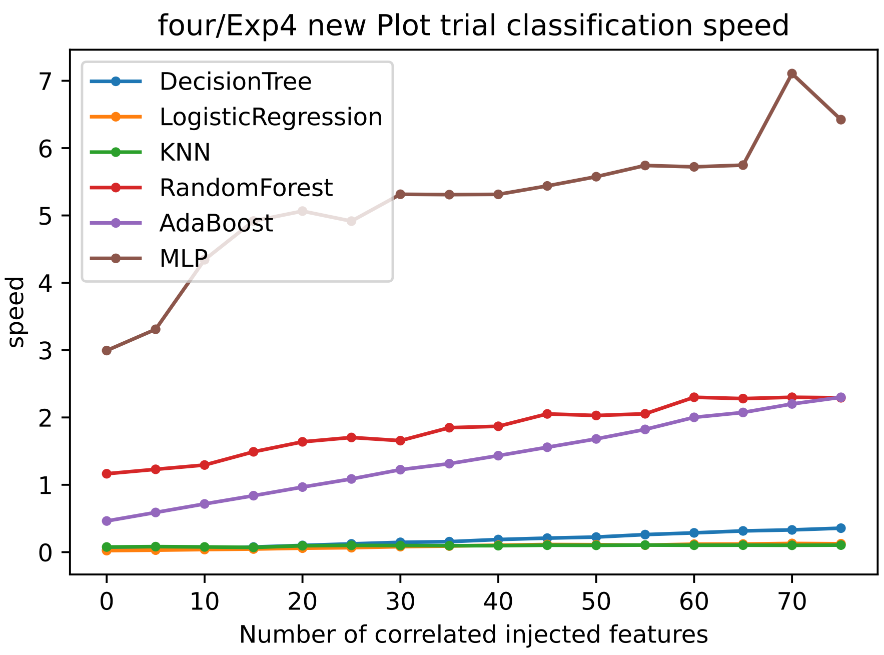

Figure 3.12: Exp4 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0290	0.0225	0.0765	1.1642	0.4624	2.9937
5	0.0435	0.0297	0.0819	1.2298	0.5892	3.3092
10	0.0585	0.0385	0.0773	1.2938	0.7161	4.3425
15	0.0760	0.0462	0.0702	1.4900	0.8384	4.9174
20	0.0998	0.0589	0.0949	1.6385	0.9663	5.0644
25	0.1230	0.0661	0.1005	1.7028	1.0871	4.9155
30	0.1457	0.0812	0.0976	1.6559	1.2246	5.3145
35	0.1556	0.0906	0.0955	1.8482	1.3138	5.3093
40	0.1873	0.1028	0.0961	1.8689	1.4328	5.3131
45	0.2082	0.1132	0.1028	2.0528	1.5574	5.4386
50	0.2239	0.1105	0.1011	2.0293	1.6814	5.5756
55	0.2606	0.1037	0.1055	2.0545	1.8225	5.7423
60	0.2862	0.1183	0.1024	2.2998	2.0017	5.7217
65	0.3152	0.1203	0.1034	2.2803	2.0739	5.7473
70	0.3302	0.1290	0.1017	2.3001	2.2007	7.1071
75	0.3559	0.1244	0.1045	2.2927	2.2994	6.4227

Table 3.12: Exp 4 Speed

3.0.5. Results experiment 5

Introduce a growing number of features with a high correlation.

In Figure 3.13 it is shown the difference between f1 train and f1 test. The general trend across models is a less pronounced increase in the gap between training and test performance compared to when low or medium correlated features are added. This can happen because high correlation reduces the effective dimensionality of the feature space since the new features don't add much new information; they are almost redundant. Thus, the models are not learning complex representations that would overfit to the training data. With high-correlated features, the models' learning process may stabilize quickly, leading to less variance between training and testing as the new features don't significantly alter the decision boundaries learned from the original features.

In this plot is possible to see that for KNN model and Random Forest the distance remains almost constant. Also for Logistic Regression there is an increase but it is way smaller than the ones showed in experiment 3 and 4. For MLP there is an increase but it is way smaller than the one showed in experiment 3 and 4.

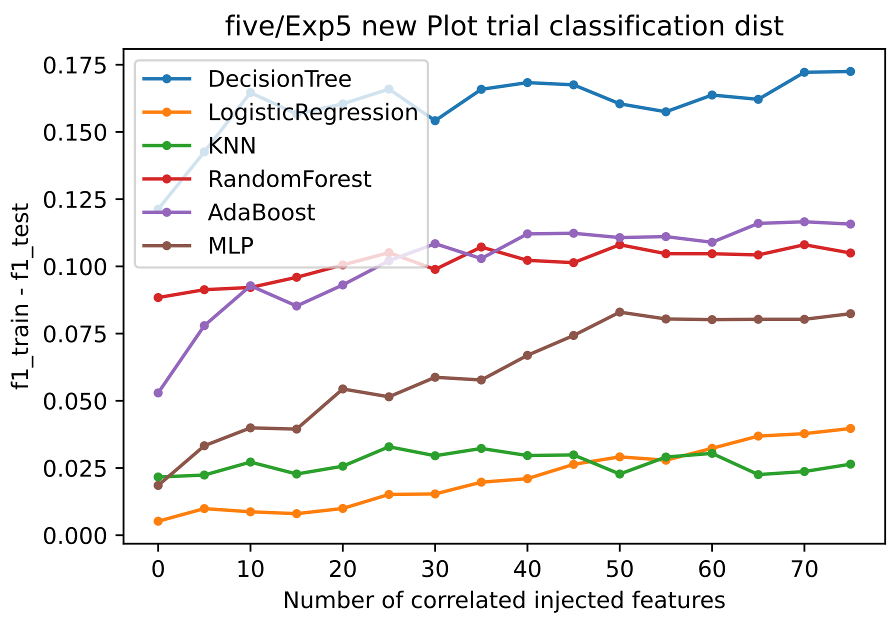

Figure 3.13: Exp5 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1213	0.0052	0.0217	0.0884	0.0529	0.0185
5	0.1426	0.0099	0.0224	0.0913	0.0780	0.0333
10	0.1646	0.0087	0.0272	0.0922	0.0928	0.0399
15	0.1567	0.0080	0.0228	0.0959	0.0853	0.0395
20	0.1604	0.0100	0.0257	0.1005	0.0931	0.0544
25	0.1660	0.0152	0.0329	0.1051	0.1021	0.0515
30	0.1542	0.0154	0.0296	0.0989	0.1084	0.0587
35	0.1659	0.0197	0.0323	0.1072	0.1029	0.0577
40	0.1684	0.0210	0.0296	0.1022	0.1121	0.0669
45	0.1675	0.0264	0.0299	0.1014	0.1123	0.0743
50	0.1605	0.0292	0.0228	0.1081	0.1107	0.0830
55	0.1575	0.0279	0.0291	0.1047	0.1111	0.0804
60	0.1637	0.0323	0.0304	0.1047	0.1090	0.0802
65	0.1621	0.0369	0.0226	0.1043	0.1160	0.0803
70	0.1722	0.0378	0.0237	0.1081	0.1166	0.0803
75	0.1725	0.0397	0.0264	0.1050	0.1157	0.0824

Table 3.13: Exp 5 Distance

In Figure 3.14 it is shown the f1 weighted score for each model on each dataset.

In the plot showing F1 weighted scores for various models with high-correlation injected features, the drop in performance is smaller than with low or medium correlation because.

High correlation means new features provide similar information to existing ones, causing less confusion to the models.

Models like **Random Forest** and **AdaBoost**, which are robust to feature redundancy due to their ensemble nature, maintain performance levels as they can disregard redundant or less informative features. In the experiment 3 and 4, the models are more likely to overfit to noise, where the increase in unrelated information might lead the models to learn spurious patterns not applicable to the test data.

For KNN, the absence of a significant drop in performance with high-correlated features may be due to its dependency on distance calculations between instances. Since the added features are highly correlated, they do not drastically change the relative distances among points, allowing KNN to maintain consistent performance.

For MLP, the resilience could be due to its ability to learn complex representations. When high-correlated features are introduced, the MLP might simply learn to give less weight to redundant information, somewhat similar to how it could learn to ignore noise. High correlation doesn't introduce as much complexity to the decision boundaries as uncorrelated features might, so the model's performance remains more stable.

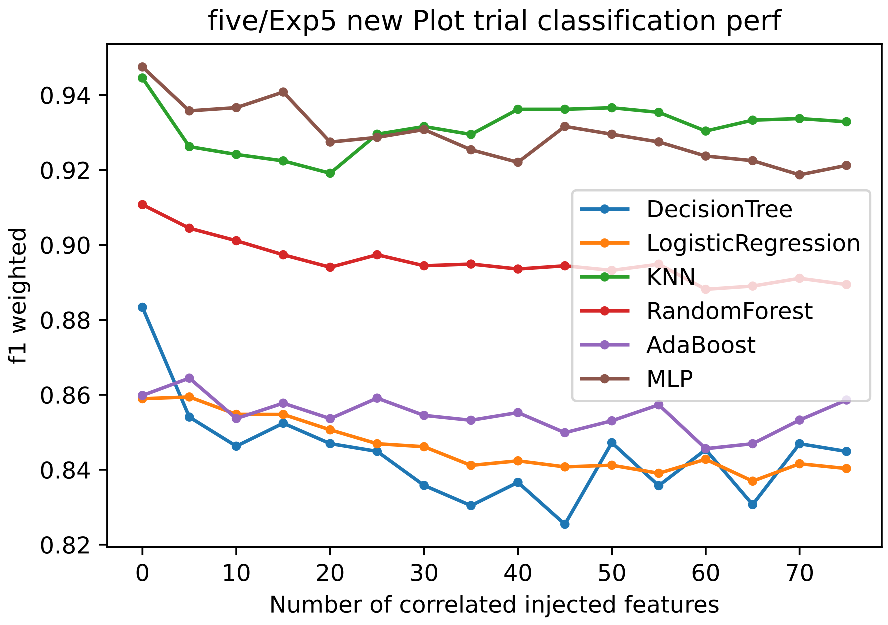

Figure 3.14: Exp5 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8833	0.8589	0.9445	0.9107	0.8598	0.9475
5	0.8541	0.8594	0.9262	0.9045	0.8644	0.9358
10	0.8463	0.8548	0.9241	0.9011	0.8537	0.9366
15	0.8524	0.8548	0.9224	0.8974	0.8578	0.9408
20	0.8470	0.8506	0.9191	0.8940	0.8536	0.9274
25	0.8449	0.8469	0.9295	0.8974	0.8591	0.9287
30	0.8358	0.8461	0.9316	0.8944	0.8545	0.9308
35	0.8304	0.8412	0.9295	0.8949	0.8532	0.9254
40	0.8366	0.8424	0.9362	0.8936	0.8552	0.9220
45	0.8254	0.8408	0.9362	0.8944	0.8499	0.9316
50	0.8472	0.8412	0.9366	0.8932	0.8531	0.9295
55	0.8357	0.8391	0.9354	0.8948	0.8573	0.9275
60	0.8454	0.8428	0.9304	0.8881	0.8456	0.9237
65	0.8307	0.8369	0.9333	0.8890	0.8469	0.9225
70	0.8469	0.8416	0.9337	0.8911	0.8533	0.9187
75	0.8449	0.8403	0.9329	0.8894	0.8586	0.9212

Table 3.14: Exp 5 Mean Performance

The explanation of the plot in Figure 3.15 is the same as the one for experiment 3. **Click here to go to the explanation**

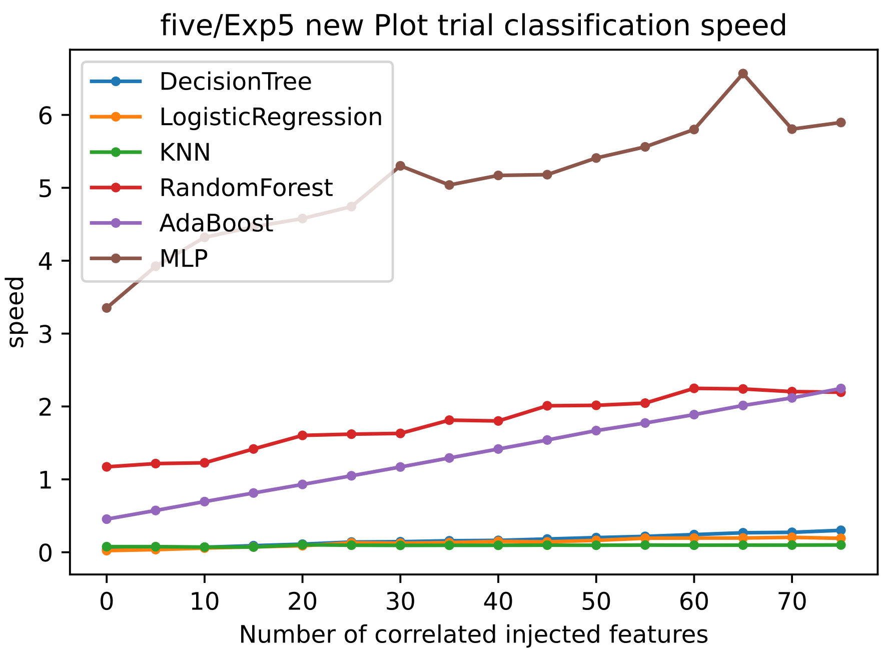

Figure 3.15: Exp5 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0350	0.0224	0.0769	1.1719	0.4544	3.3518
5	0.0542	0.0363	0.0770	1.2165	0.5734	3.9237
10	0.0686	0.0574	0.0708	1.2274	0.6950	4.3203
15	0.0905	0.0739	0.0715	1.4171	0.8131	4.4645
20	0.1111	0.0893	0.1026	1.6031	0.9306	4.5782
25	0.1404	0.1299	0.0969	1.6204	1.0492	4.7423
30	0.1448	0.1246	0.0946	1.6305	1.1699	5.3017
35	0.1577	0.1324	0.0955	1.8126	1.2937	5.0385
40	0.1631	0.1494	0.0955	1.8013	1.4166	5.1700
45	0.1819	0.1423	0.0975	2.0096	1.5401	5.1804
50	0.2010	0.1626	0.0961	2.0160	1.6692	5.4087
55	0.2180	0.1931	0.0990	2.0460	1.7732	5.5620
60	0.2425	0.1950	0.0970	2.2487	1.8884	5.7999
65	0.2675	0.1956	0.0983	2.2406	2.0139	6.5673
70	0.2737	0.2047	0.0986	2.2039	2.1177	5.8051
75	0.3009	0.1914	0.0999	2.1957	2.2469	5.8964

Table 3.15: Exp 5 Speed

3.0.6. Results experiment 6

Introduce a growing number of features with a negative correlation.

In Figure 3.16 it is shown the distance between the f1 train and the f1 test. It is possible to see that the results are really similar to the ones obtained in the experiment 4 in which the correlation strength is similar with the difference that in experiment 4 the correlation is positive and in the experiment 6 it is negative.

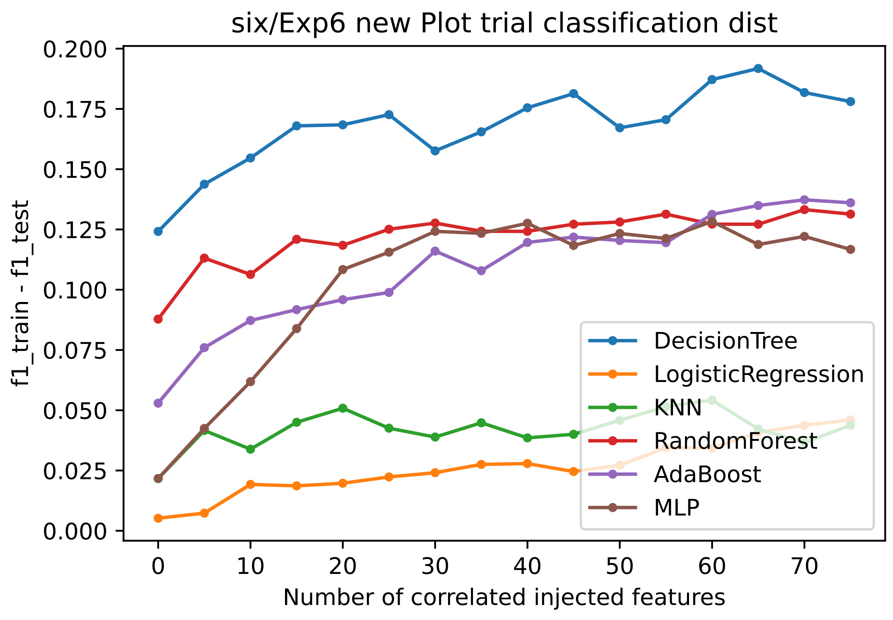

Figure 3.16: Exp6 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1242	0.0052	0.0217	0.0878	0.0529	0.0217
5	0.1438	0.0073	0.0416	0.1131	0.0760	0.0424
10	0.1547	0.0192	0.0338	0.1063	0.0872	0.0619
15	0.1680	0.0186	0.0450	0.1209	0.0917	0.0839
20	0.1684	0.0197	0.0508	0.1184	0.0959	0.1084
25	0.1726	0.0223	0.0426	0.1251	0.0989	0.1156
30	0.1576	0.0241	0.0389	0.1277	0.1160	0.1242
35	0.1655	0.0275	0.0448	0.1243	0.1079	0.1234
40	0.1755	0.0279	0.0385	0.1242	0.1196	0.1276
45	0.1814	0.0246	0.0400	0.1272	0.1218	0.1184
50	0.1672	0.0272	0.0458	0.1281	0.1204	0.1233
55	0.1705	0.0345	0.0515	0.1314	0.1195	0.1213
60	0.1872	0.0342	0.0542	0.1272	0.1312	0.1283
65	0.1918	0.0408	0.0422	0.1272	0.1349	0.1188
70	0.1818	0.0438	0.0366	0.1332	0.1373	0.1221
75	0.1781	0.0460	0.0438	0.1314	0.1361	0.1167

Table 3.16: Exp 6 Distance

In Figure 3.17 it is shown the F1 weighted score for each model on each dataset. The result are similar and comparable to the ones obtained in experiment 4, showing that there is not much difference in having a negative correlation or a positive correlation if the correlation strength is the same in absolute value.

The only thing is that MLP is performing a little bit better in experiment 4 and KNN the opposite.

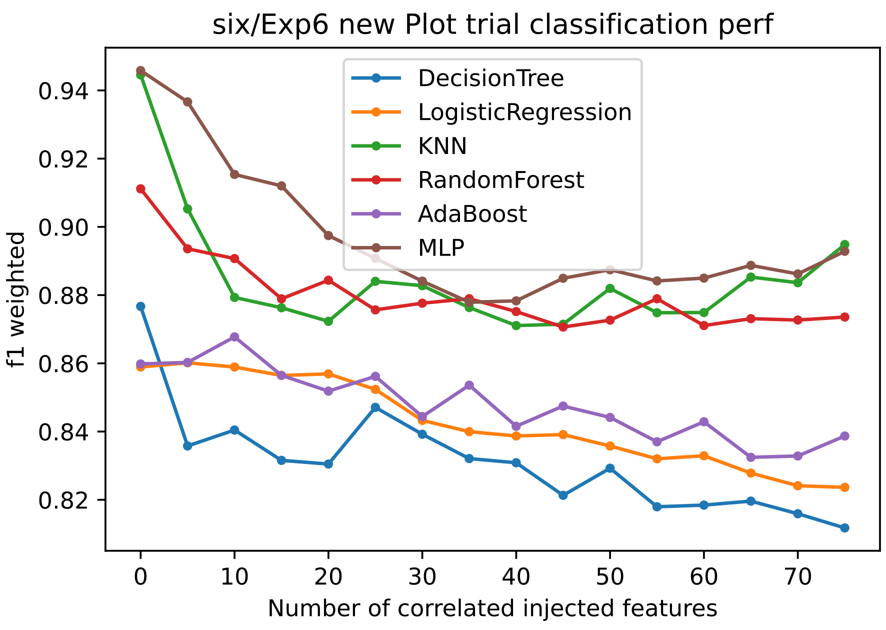

Figure 3.17: Exp6 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8767	0.8589	0.9445	0.9112	0.8598	0.9458
5	0.8358	0.8602	0.9053	0.8936	0.8603	0.9366
10	0.8405	0.8589	0.8793	0.8907	0.8678	0.9154
15	0.8316	0.8564	0.8763	0.8789	0.8565	0.9120
20	0.8305	0.8569	0.8723	0.8844	0.8519	0.8975
25	0.8471	0.8524	0.8840	0.8757	0.8562	0.8908
30	0.8392	0.8433	0.8828	0.8776	0.8444	0.8841
35	0.8321	0.8400	0.8764	0.8789	0.8536	0.8779
40	0.8309	0.8387	0.8711	0.8752	0.8416	0.8783
45	0.8213	0.8391	0.8715	0.8706	0.8475	0.8849
50	0.8293	0.8358	0.8820	0.8727	0.8441	0.8874
55	0.8180	0.8320	0.8748	0.8789	0.8370	0.8842
60	0.8184	0.8329	0.8749	0.8711	0.8429	0.8850
65	0.8196	0.8278	0.8853	0.8731	0.8324	0.8887
70	0.8159	0.8241	0.8836	0.8727	0.8328	0.8862
75	0.8118	0.8236	0.8948	0.8736	0.8387	0.8928

Table 3.17: Exp 6 Mean Performance

The explanation of the plot in Figure 3.18 is the same as the one for experiment 3. **Click here to go to the explanation**

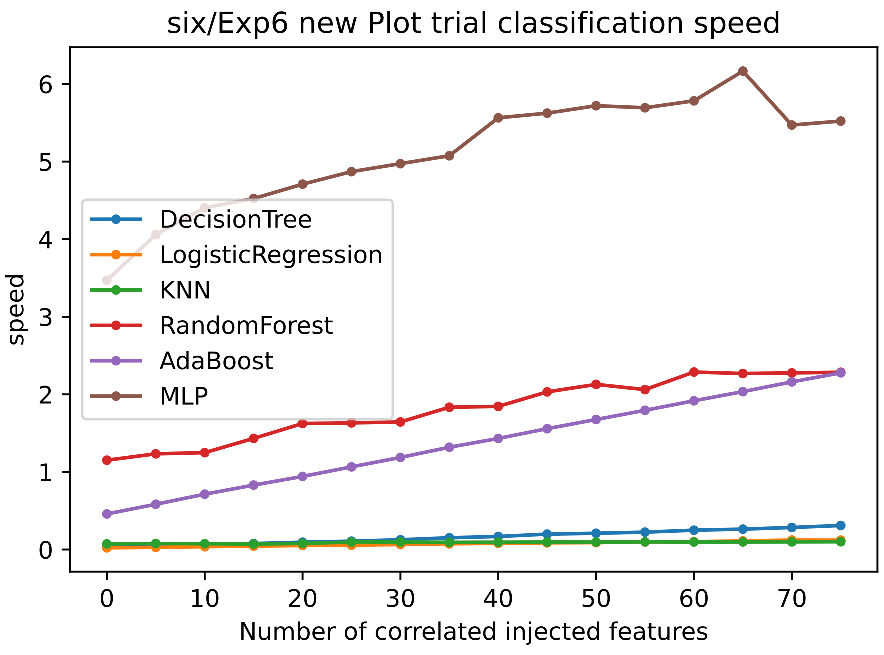

Figure 3.18: Exp6 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0288	0.0220	0.0726	1.1513	0.4585	3.4700
5	0.0438	0.0278	0.0776	1.2335	0.5833	4.0586
10	0.0604	0.0367	0.0751	1.2484	0.7131	4.4054
15	0.0770	0.0438	0.0697	1.4323	0.8302	4.5237
20	0.0941	0.0518	0.0789	1.6242	0.9421	4.7090
25	0.1075	0.0562	0.0960	1.6319	1.0652	4.8706
30	0.1256	0.0644	0.0973	1.6443	1.1868	4.9734
35	0.1514	0.0737	0.0927	1.8338	1.3182	5.0752
40	0.1681	0.0797	0.0942	1.8449	1.4317	5.5642
45	0.1985	0.0860	0.0961	2.0319	1.5576	5.6244
50	0.2096	0.0890	0.0958	2.1282	1.6758	5.7201
55	0.2239	0.0978	0.0987	2.0613	1.7941	5.6937
60	0.2492	0.1010	0.0973	2.2875	1.9171	5.7831
65	0.2628	0.1107	0.0978	2.2686	2.0351	6.1662
70	0.2840	0.1219	0.0986	2.2765	2.1605	5.4710
75	0.3098	0.1204	0.1002	2.2847	2.2772	5.5221

Table 3.18: Exp 6 Speed

3.0.7. Results experiment 7

Inject more and more non linear correlated features (power of 2 of a feature).

In Figure 3.19 it is shown the distance f1 train and f1 test.

The behavior is pretty similar to the experiment 3 and 4 even though the type of feature injected is different.

For **Random Forest** the difference remains almost constant.

Instead for **MLP** the gap increases significantly with the introduction of non-linear features, indicating potential overfitting as MLPs, with their capacity for modeling complex relationships, might be capturing the noise in the training data. But this increase is gradual and it is different with respect to experiment 3 and experiment 4: in those cases the increase was initially way faster but then it slowed down and the distance was bigger too.

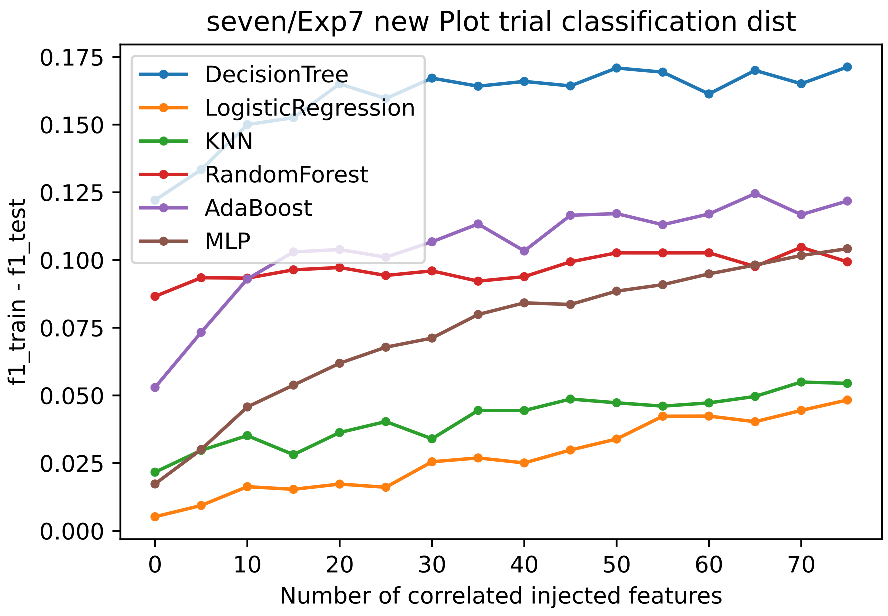

Figure 3.19: Exp7 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1221	0.0052	0.0217	0.0866	0.0529	0.0173
5	0.1334	0.0094	0.0297	0.0935	0.0733	0.0300
10	0.1500	0.0163	0.0352	0.0933	0.0930	0.0457
15	0.1526	0.0153	0.0281	0.0964	0.1030	0.0538
20	0.1650	0.0173	0.0363	0.0972	0.1039	0.0619
25	0.1596	0.0161	0.0403	0.0943	0.1011	0.0678
30	0.1672	0.0255	0.0340	0.0960	0.1067	0.0712
35	0.1642	0.0269	0.0445	0.0922	0.1133	0.0799
40	0.1659	0.0251	0.0444	0.0939	0.1033	0.0842
45	0.1643	0.0299	0.0487	0.0993	0.1165	0.0836
50	0.1709	0.0339	0.0473	0.1026	0.1171	0.0885
55	0.1694	0.0423	0.0460	0.1026	0.1130	0.0909
60	0.1613	0.0424	0.0472	0.1027	0.1170	0.0949
65	0.1700	0.0403	0.0496	0.0976	0.1245	0.0982
70	0.1651	0.0445	0.0549	0.1047	0.1168	0.1017
75	0.1713	0.0483	0.0545	0.0993	0.1218	0.1042

Table 3.19: Exp 7 Distance

In Figure 3.20 it is shown the F1 weighted score for each model on each dataset.

The most interesting result for this experiment is that there is a slightly increase in performance of the **Logistic Regression** model, when non-linearly correlated features are added. Even though Logistic Regression is a linear model, the squaring of features might create a scenario where the transformed features align better with the linear decision boundary the model is trying to learn.

KNN shows a decreasing trend, that is caused by the introduction of these new type of non-linear correlated features.

AdaBoost shows variability in performance, but no big drop which shows that it is overall quite a robust model and also **Random Forest** model being an ensemble model remains pretty stable.

MLP exhibits a drop and a trend of decreasing f1, suggesting that while MLPs can model non-linear relationships, the added complexity from the new features may lead to overfitting or difficulty in training.

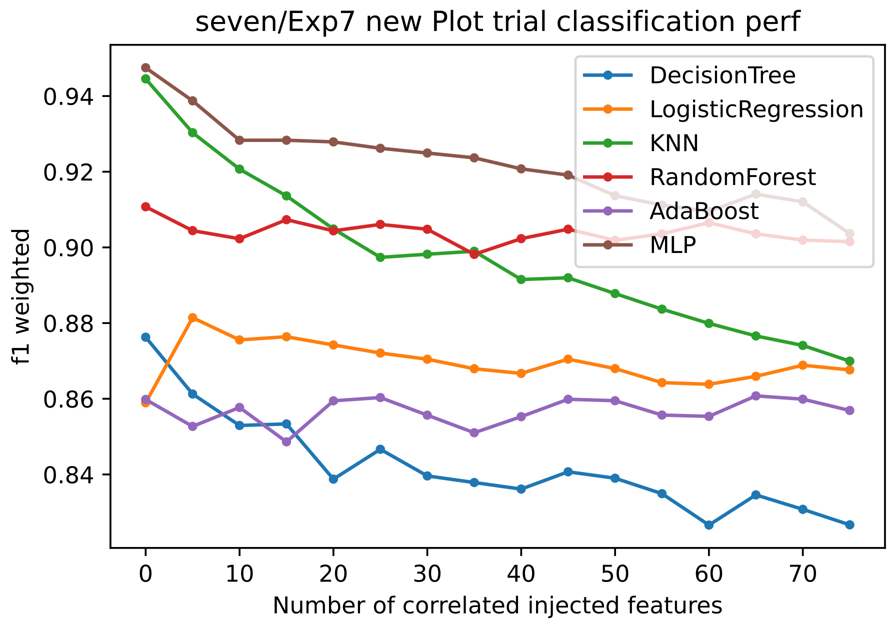

Figure 3.20: Exp7 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8763	0.8589	0.9445	0.9107	0.8598	0.9475
5	0.8613	0.8814	0.9303	0.9044	0.8527	0.9387
10	0.8529	0.8756	0.9207	0.9023	0.8577	0.9283
15	0.8533	0.8764	0.9136	0.9073	0.8486	0.9283
20	0.8388	0.8742	0.9049	0.9044	0.8595	0.9279
25	0.8467	0.8721	0.8974	0.9061	0.8603	0.9262
30	0.8396	0.8705	0.8982	0.9048	0.8557	0.9249
35	0.8379	0.8679	0.8990	0.8982	0.8510	0.9237
40	0.8361	0.8667	0.8915	0.9023	0.8553	0.9208
45	0.8407	0.8705	0.8920	0.9048	0.8599	0.9191
50	0.8390	0.8680	0.8878	0.9019	0.8595	0.9137
55	0.8349	0.8643	0.8837	0.9036	0.8557	0.9112
60	0.8266	0.8638	0.8799	0.9065	0.8553	0.9095
65	0.8346	0.8659	0.8766	0.9036	0.8608	0.9141
70	0.8308	0.8689	0.8741	0.9019	0.8599	0.9120
75	0.8267	0.8676	0.8699	0.9015	0.8569	0.9037

Table 3.20: Exp 7 Mean Performance

The explanation of the plot in Figure 3.21 is the same as the one for experiment 3. **Click here to go to the explanation**

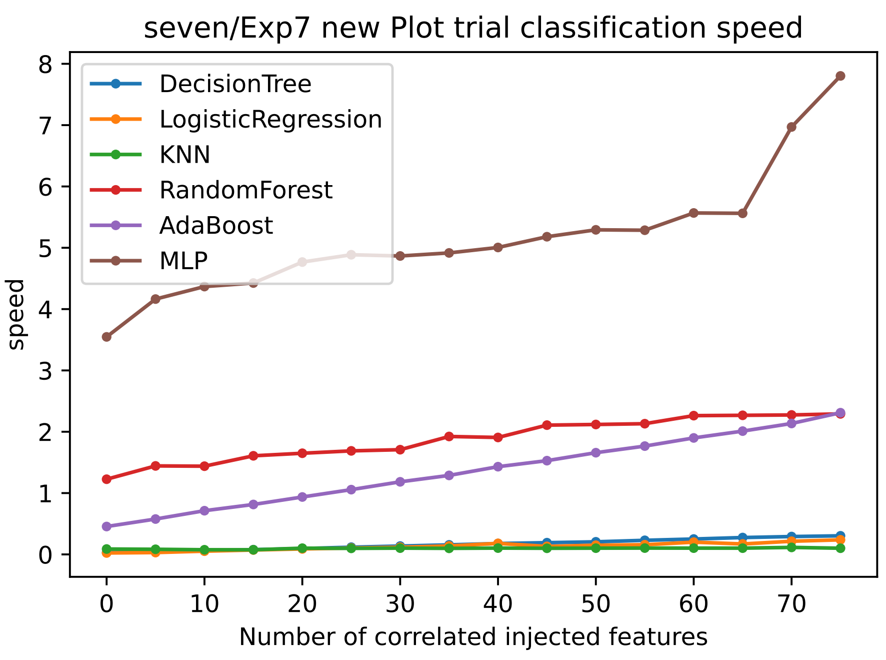

Figure 3.21: Exp7 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0291	0.0233	0.0879	1.2260	0.4547	3.5469
5	0.0413	0.0301	0.0834	1.4430	0.5767	4.1635
10	0.0596	0.0527	0.0758	1.4384	0.7126	4.3675
15	0.0774	0.0717	0.0755	1.6078	0.8141	4.4250
20	0.0954	0.0894	0.1014	1.6495	0.9362	4.7670
25	0.1177	0.1046	0.0986	1.6883	1.0562	4.8862
30	0.1361	0.1171	0.1024	1.7076	1.1836	4.8661
35	0.1556	0.1379	0.0989	1.9231	1.2881	4.9162
40	0.1776	0.1799	0.1031	1.9070	1.4305	5.0051
45	0.1914	0.1305	0.1001	2.1081	1.5286	5.1802
50	0.2054	0.1493	0.1020	2.1186	1.6585	5.2926
55	0.2298	0.1576	0.1034	2.1313	1.7661	5.2863
60	0.2510	0.1998	0.1024	2.2629	1.8993	5.5676
65	0.2749	0.1703	0.1020	2.2681	2.0105	5.5620
70	0.2911	0.2144	0.1141	2.2741	2.1348	6.9698
75	0.3032	0.2375	0.1008	2.2916	2.3104	7.8025

Table 3.21: Exp 7 Speed

3.0.8. Results experiment 8

Inject more and more non linear correlated features (cosine).

The slightly larger gap when using cosine-based non-linear features, compared to squared features, could be because cosine transformations can create more complex and periodic relationships that do not align well with the assumptions of many learning algorithms. Cosine transformations, however, can introduce cyclical patterns that are harder to model, potentially leading to models that fit the training data well but perform poorly on unseen test data, hence a larger gap between training and testing performance.

Some models like **MLPs** are sensitive to these patterns and may fit the training data closely, but fail to predict unseen data as accurately.

For **KNN**, the gap between training and testing performance increasing significantly with the introduction of cosine-based non-linear features could be because KNN relies heavily on the distance between points to make predictions.

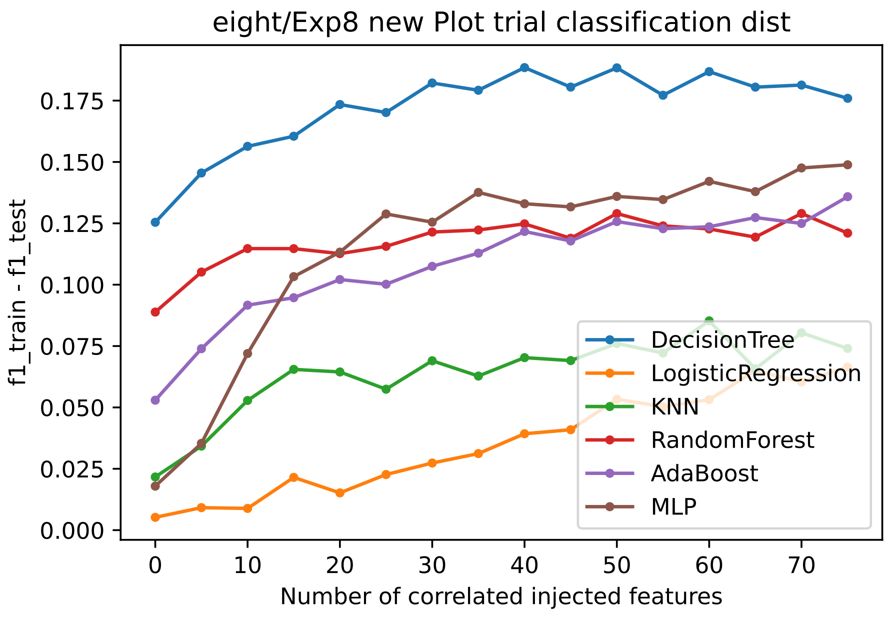

Figure 3.22: Exp8 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1254	0.0052	0.0217	0.0888	0.0529	0.0179
5	0.1455	0.0092	0.0342	0.1052	0.0739	0.0354
10	0.1564	0.0089	0.0528	0.1147	0.0917	0.0720
15	0.1605	0.0215	0.0655	0.1147	0.0947	0.1033
20	0.1734	0.0152	0.0645	0.1127	0.1021	0.1133
25	0.1702	0.0227	0.0574	0.1156	0.1002	0.1288
30	0.1822	0.0273	0.0690	0.1214	0.1075	0.1255
35	0.1792	0.0312	0.0628	0.1223	0.1128	0.1376
40	0.1884	0.0393	0.0703	0.1248	0.1217	0.1330
45	0.1805	0.0409	0.0691	0.1189	0.1178	0.1317
50	0.1883	0.0533	0.0760	0.1290	0.1257	0.1360
55	0.1772	0.0503	0.0722	0.1240	0.1228	0.1347
60	0.1868	0.0532	0.0853	0.1227	0.1236	0.1421
65	0.1805	0.0650	0.0657	0.1194	0.1274	0.1379
70	0.1813	0.0604	0.0804	0.1290	0.1250	0.1476
75	0.1759	0.0664	0.0740	0.1210	0.1359	0.1489

Table 3.22: Exp 8 Distance

In Figure 3.23 it is shown the F1 weighted score of each model on each dataset.

Similarly to what was achieved during experiment 7, Logistic Regression is having a slightly increase in performance.

It is interesting to notice the big drop that **MLP** has. It can be partially explained with the plot in Figure 3.22 in which it is clear that the model is overfitting.

However the biggest drop in performance is recorded by **KNN**. The algorithm strongly relies on the distance between points and the introduction of this new type of feature is causing a lot of damage to the capability of the model, making him the worst model in this experiment.

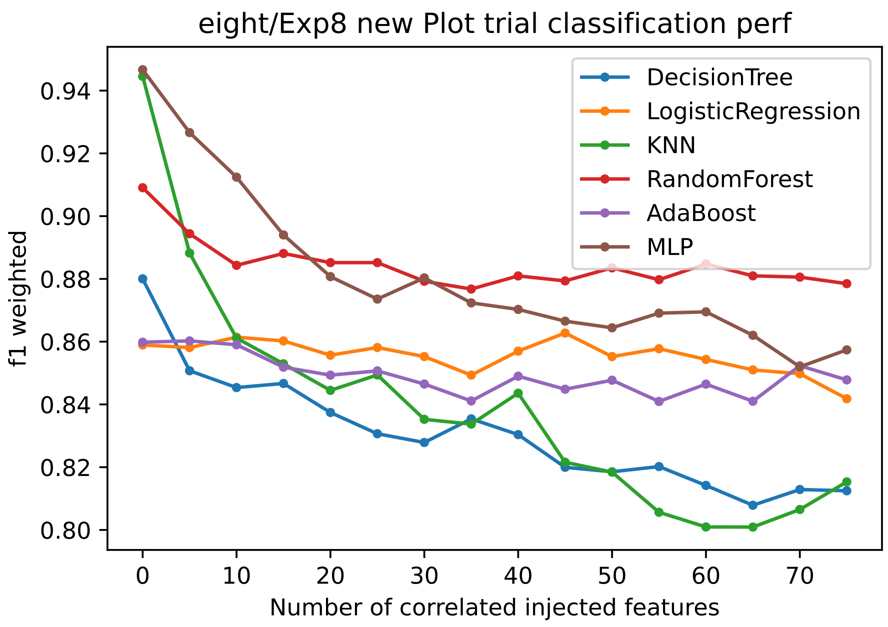

Figure 3.23: Exp8 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8800	0.8589	0.9445	0.9091	0.8598	0.9467
5	0.8507	0.8581	0.8882	0.8944	0.8602	0.9266
10	0.8454	0.8614	0.8611	0.8843	0.8590	0.9124
15	0.8467	0.8602	0.8529	0.8881	0.8519	0.8940
20	0.8374	0.8557	0.8445	0.8852	0.8493	0.8807
25	0.8307	0.8581	0.8494	0.8852	0.8507	0.8736
30	0.8279	0.8552	0.8352	0.8792	0.8465	0.8803
35	0.8354	0.8493	0.8337	0.8768	0.8411	0.8724
40	0.8304	0.8570	0.8436	0.8809	0.8490	0.8703
45	0.8200	0.8628	0.8216	0.8794	0.8448	0.8665
50	0.8185	0.8552	0.8184	0.8835	0.8477	0.8644
55	0.8202	0.8578	0.8056	0.8797	0.8409	0.8691
60	0.8142	0.8543	0.8009	0.8848	0.8465	0.8695
65	0.8079	0.8510	0.8009	0.8810	0.8410	0.8620
70	0.8129	0.8498	0.8065	0.8806	0.8523	0.8520
75	0.8125	0.8418	0.8153	0.8785	0.8478	0.8574

Table 3.23: Exp 8 Mean Performance

The explanation of the plot in Figure 3.24 is the same as the one for experiment 3. **Click here to go to the explanation**

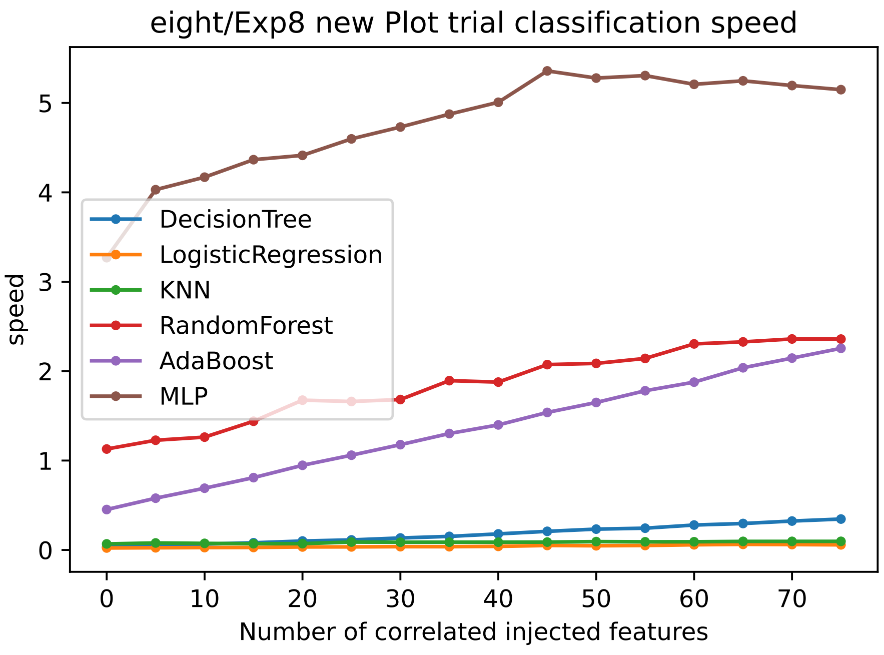

Figure 3.24: Exp8 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0312	0.0221	0.0677	1.1290	0.4520	3.2694
5	0.0512	0.0243	0.0782	1.2272	0.5789	4.0296
10	0.0656	0.0264	0.0736	1.2620	0.6906	4.1696
15	0.0803	0.0278	0.0699	1.4395	0.8089	4.3655
20	0.1001	0.0336	0.0705	1.6755	0.9468	4.4137
25	0.1123	0.0342	0.0885	1.6606	1.0598	4.5980
30	0.1340	0.0367	0.0862	1.6824	1.1778	4.7307
35	0.1515	0.0364	0.0867	1.8940	1.3023	4.8744
40	0.1795	0.0402	0.0870	1.8775	1.3987	5.0076
45	0.2083	0.0499	0.0881	2.0733	1.5381	5.3583
50	0.2330	0.0468	0.0933	2.0867	1.6495	5.2783
55	0.2434	0.0498	0.0912	2.1421	1.7807	5.3061
60	0.2787	0.0578	0.0916	2.3049	1.8771	5.2084
65	0.2957	0.0625	0.0957	2.3273	2.0377	5.2475
70	0.3233	0.0602	0.0959	2.3600	2.1457	5.1947
75	0.3457	0.0572	0.0962	2.3586	2.2552	5.1486

Table 3.24: Exp 8 Speed

3.0.9. Results experiment 9

Introducing new features as a linear combination of existing features.

In Figure 3.25 it is shown the distance between the F1 train and F1 test. There are not significant changes and that could be explained by the fact that only few features were injected.

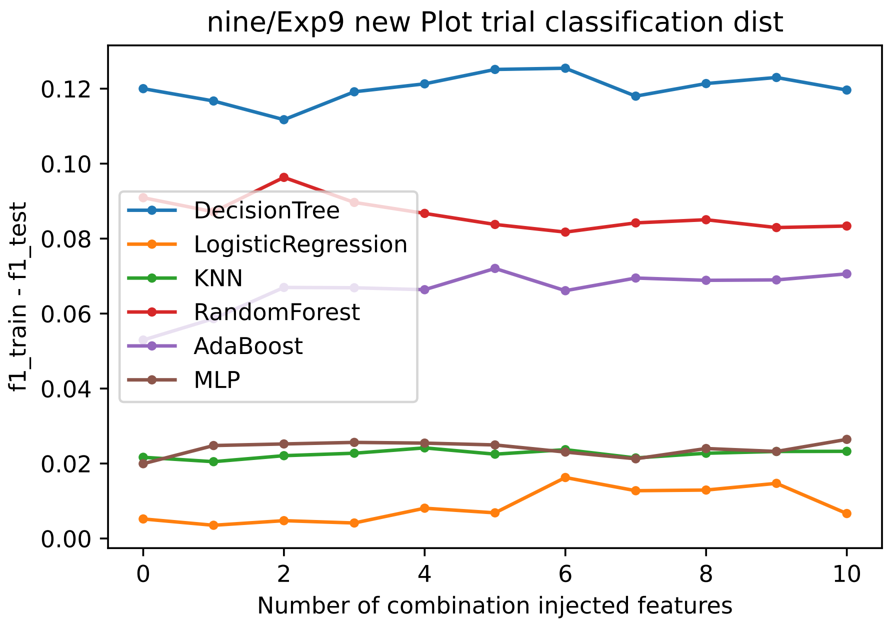

Figure 3.25: Exp9 distance between train and test performances

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.1200	0.0052	0.0217	0.0909	0.0529	0.0199
1	0.1167	0.0035	0.0205	0.0871	0.0587	0.0248
2	0.1117	0.0048	0.0221	0.0963	0.0670	0.0252
3	0.1192	0.0041	0.0227	0.0897	0.0669	0.0256
4	0.1213	0.0081	0.0242	0.0867	0.0664	0.0254
5	0.1251	0.0068	0.0225	0.0838	0.0720	0.0250
6	0.1255	0.0163	0.0237	0.0817	0.0661	0.0231
7	0.1180	0.0127	0.0215	0.0842	0.0695	0.0213
8	0.1214	0.0129	0.0227	0.0850	0.0689	0.0240
9	0.1230	0.0147	0.0232	0.0829	0.0690	0.0232
10	0.1196	0.0067	0.0233	0.0834	0.0706	0.0264

Table 3.25: Exp 9 Distance

In Figure 3.26 it is shown the F1 weighted score for each model on each dataset.

The results of this experiment are interesting because it is possible to see that with the exception of KNN, the other models tends to remain constant or improve performances.

Some fluctuations in the performance could be caused by the way the data is split into train-val-test set.

But it is important to notice that for Logistic Regression and also AdaBoost it is possible to see improvements as new features are inserted. Sometimes when performing Feature Engineering it may be good to insert some features as the result of the combination of existing features to try to capture more complex relationships and improve a model's ability to capture nonlinear patterns in the data.

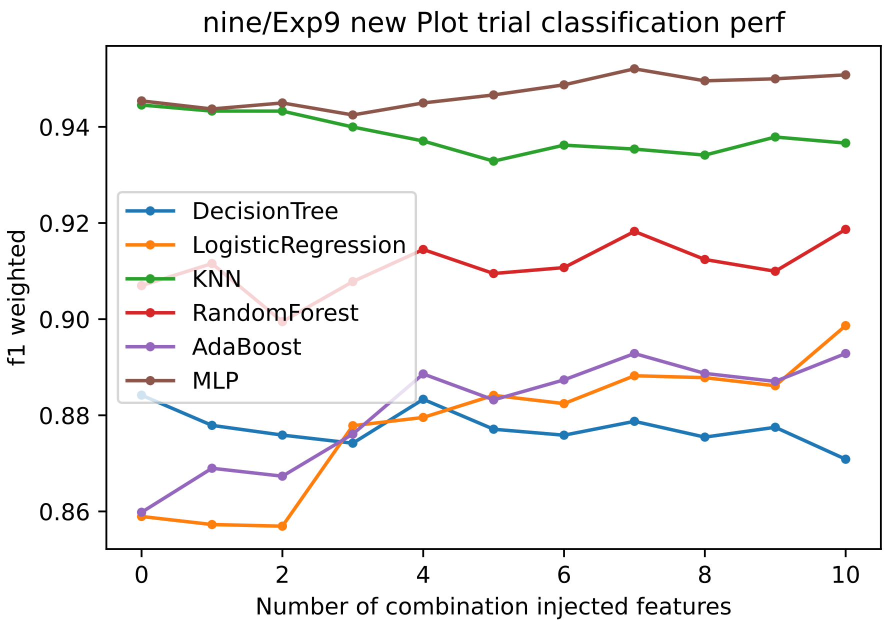

Figure 3.26: Exp9 F1 weighted score

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.8842	0.8589	0.9445	0.9070	0.8598	0.9454
1	0.8779	0.8573	0.9433	0.9116	0.8690	0.9437
2	0.8759	0.8569	0.9433	0.8995	0.8673	0.9450
3	0.8742	0.8778	0.9400	0.9078	0.8761	0.9425
4	0.8833	0.8795	0.9371	0.9145	0.8886	0.9450
5	0.8771	0.8841	0.9329	0.9095	0.8832	0.9466
6	0.8759	0.8824	0.9362	0.9107	0.8874	0.9487
7	0.8787	0.8882	0.9354	0.9183	0.8928	0.9521
8	0.8755	0.8878	0.9341	0.9124	0.8887	0.9496
9	0.8775	0.8862	0.9379	0.9100	0.8870	0.9500
10	0.8709	0.8987	0.9366	0.9187	0.8928	0.9508

Table 3.26: Exp 9 Mean Performance

In Figure 3.27 it is shown the time performance of the various models. There is a slight increase of training time as the number of injected features grows but since the number of inject features is relatively low it is not possible to see a big increment in the time to train the models, with the exception of MLP.

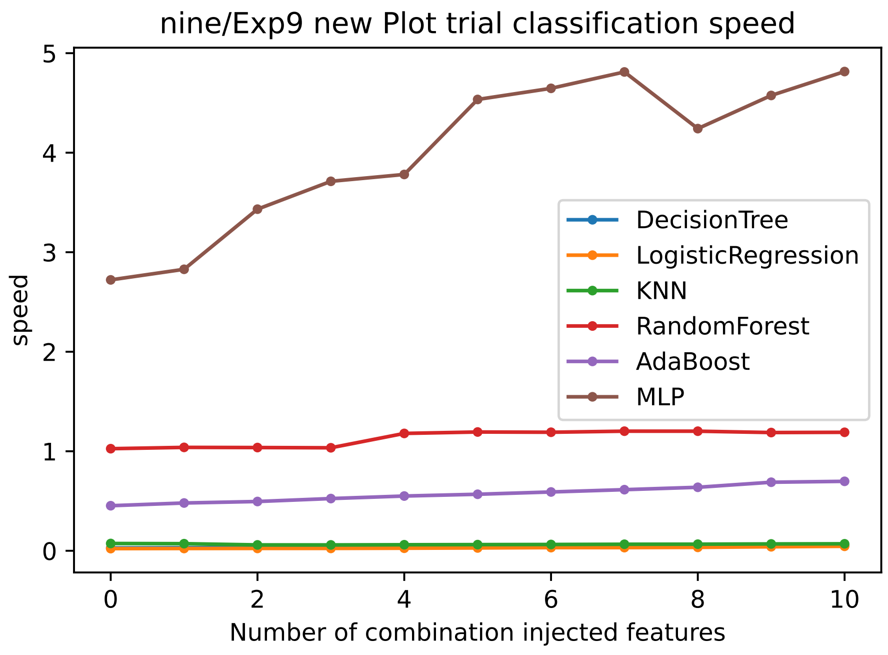

Figure 3.27: Exp9 speed of training

Injected features	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
0	0.0287	0.0219	0.0733	1.0259	0.4529	2.7220
1	0.0310	0.0231	0.0714	1.0390	0.4802	2.8284
2	0.0314	0.0238	0.0591	1.0374	0.4952	3.4325
3	0.0350	0.0239	0.0590	1.0350	0.5248	3.7123
4	0.0376	0.0257	0.0608	1.1793	0.5502	3.7813
5	0.0402	0.0284	0.0621	1.1936	0.5679	4.5349
6	0.0405	0.0320	0.0633	1.1911	0.5910	4.6466
7	0.0434	0.0320	0.0657	1.2018	0.6142	4.8113
8	0.0467	0.0345	0.0664	1.2016	0.6380	4.2420
9	0.0496	0.0403	0.0688	1.1881	0.6889	4.5751
10	0.0537	0.0451	0.0703	1.1905	0.6979	4.8153

Table 3.27: Exp 9 Speed

3.0.10. Results experiment 10

Introducing 5 correlated features going from -1 to 1 correlation strength.

In Figure 3.28 it is shown the difference between F1 train and F1 test.

In this case the gap has some oscillation as the correlation value of the inserted feature changes. Those fluctuations could be determined by the way the train-val-test split is performed.

For KNN there is a slightly increase when introducing a feature with a low correlation value.

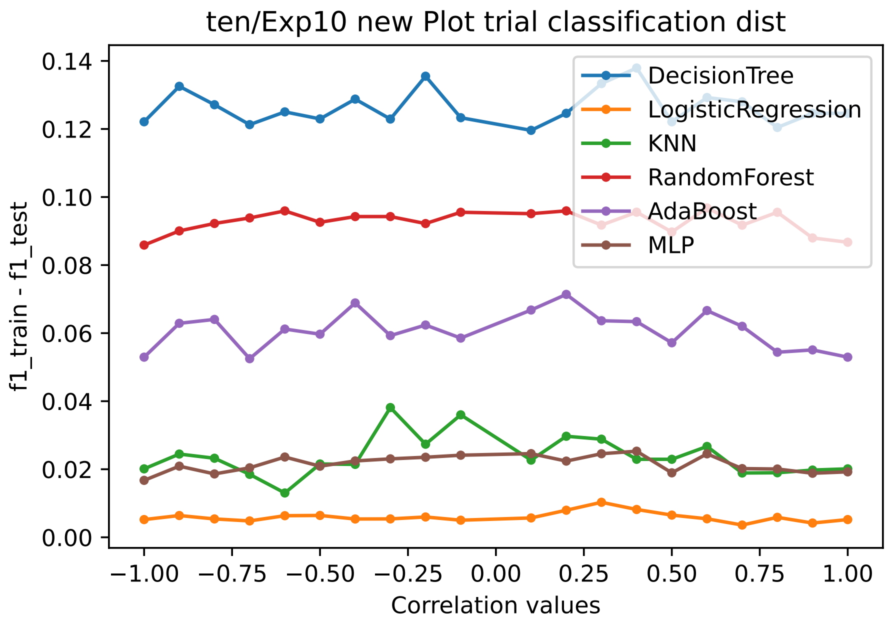

Figure 3.28: Exp10 distance between train and test performances

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
-1	0.1221	0.0052	0.0201	0.0859	0.0529	0.0167
-0.9	0.1325	0.0064	0.0245	0.0900	0.0629	0.0209
-0.8	0.1271	0.0054	0.0232	0.0922	0.0640	0.0186
-0.7	0.1213	0.0048	0.0185	0.0939	0.0525	0.0204
-0.6	0.1250	0.0063	0.0130	0.0959	0.0612	0.0236
-0.5	0.1230	0.0064	0.0215	0.0926	0.0597	0.0209
-0.4	0.1288	0.0054	0.0214	0.0943	0.0689	0.0224
-0.3	0.1229	0.0054	0.0381	0.0943	0.0593	0.0231
-0.2	0.1355	0.0060	0.0274	0.0922	0.0624	0.0235
-0.1	0.1233	0.0050	0.0360	0.0955	0.0585	0.0241
0.1	0.1196	0.0057	0.0227	0.0951	0.0668	0.0246
0.2	0.1246	0.0080	0.0297	0.0959	0.0714	0.0224
0.3	0.1333	0.0103	0.0288	0.0918	0.0636	0.0246
0.4	0.1379	0.0082	0.0229	0.0955	0.0634	0.0253
0.5	0.1221	0.0065	0.0229	0.0897	0.0572	0.0189
0.6	0.1292	0.0054	0.0267	0.0968	0.0666	0.0245
0.7	0.1280	0.0036	0.0189	0.0917	0.0620	0.0202
0.8	0.1204	0.0059	0.0190	0.0955	0.0544	0.0201
0.9	0.1246	0.0042	0.0198	0.0880	0.0551	0.0188
1	0.1246	0.0052	0.0201	0.0867	0.0529	0.0192

Table 3.28: Exp 10 Distance

In Figure 3.29 it is shown the F1 weighted score for each model on each dataset.

It is possible to see that for MLP, KNN, Random Forest and Decision Tree the performance degrades when injecting features with correlation value that ranges from -0.45 to 0.45, which is something that can be seen also in the experiment 3.

For MLP, KNN and Random Forest the performance are pretty similar in the extremes when correlation value is getting close to -1 or close to 1.

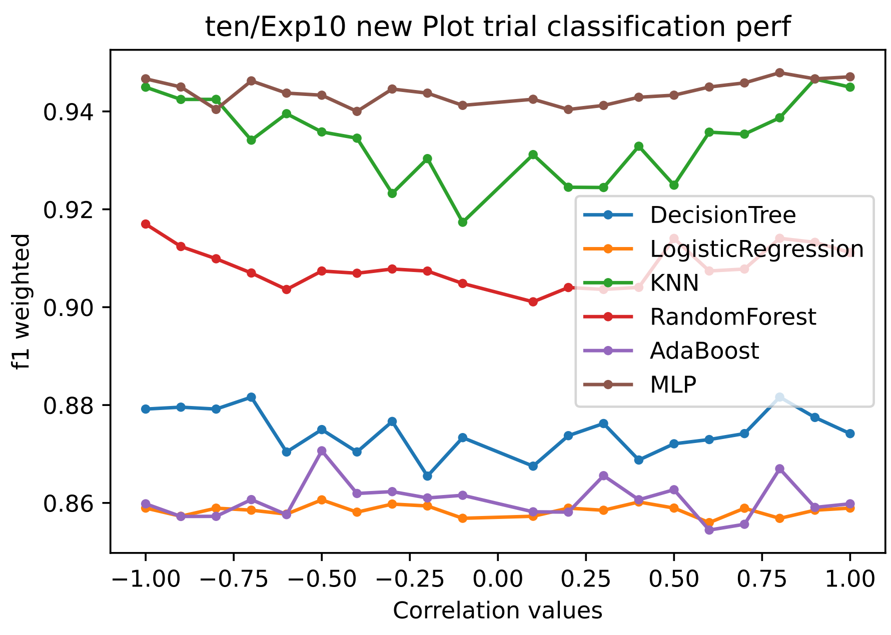

Figure 3.29: Exp10 F1 weighted score

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
-1	0.8792	0.8589	0.9450	0.9170	0.8598	0.9467
-0.9	0.8796	0.8573	0.9425	0.9124	0.8572	0.9450
-0.8	0.8792	0.8589	0.9425	0.9099	0.8573	0.9404
-0.7	0.8816	0.8585	0.9341	0.9070	0.8607	0.9462
-0.6	0.8704	0.8577	0.9395	0.9036	0.8576	0.9437
-0.5	0.8750	0.8606	0.9358	0.9074	0.8707	0.9433
-0.4	0.8704	0.8581	0.9345	0.9069	0.8619	0.9400
-0.3	0.8766	0.8598	0.9232	0.9078	0.8623	0.9446
-0.2	0.8655	0.8594	0.9304	0.9074	0.8610	0.9437
-0.1	0.8733	0.8569	0.9174	0.9048	0.8616	0.9412
0.1	0.8675	0.8573	0.9312	0.9011	0.8582	0.9425
0.2	0.8737	0.8589	0.9245	0.9040	0.8581	0.9404
0.3	0.8762	0.8585	0.9245	0.9036	0.8656	0.9412
0.4	0.8688	0.8602	0.9329	0.9040	0.8607	0.9429
0.5	0.8721	0.8589	0.9249	0.9140	0.8627	0.9433
0.6	0.8730	0.8560	0.9358	0.9074	0.8545	0.9450
0.7	0.8742	0.8589	0.9354	0.9078	0.8556	0.9458
0.8	0.8816	0.8568	0.9387	0.9141	0.8670	0.9479
0.9	0.8775	0.8585	0.9466	0.9132	0.8591	0.9467
1	0.8742	0.8589	0.9450	0.9112	0.8598	0.9471

Table 3.29: Exp 10 Mean Performance

In Figure 3.30 it is shown the time performance of the various models. The time remains almost constant for every model because for this experiment the datasets have the same number of features and they differ only by the correlation values of the injected features.

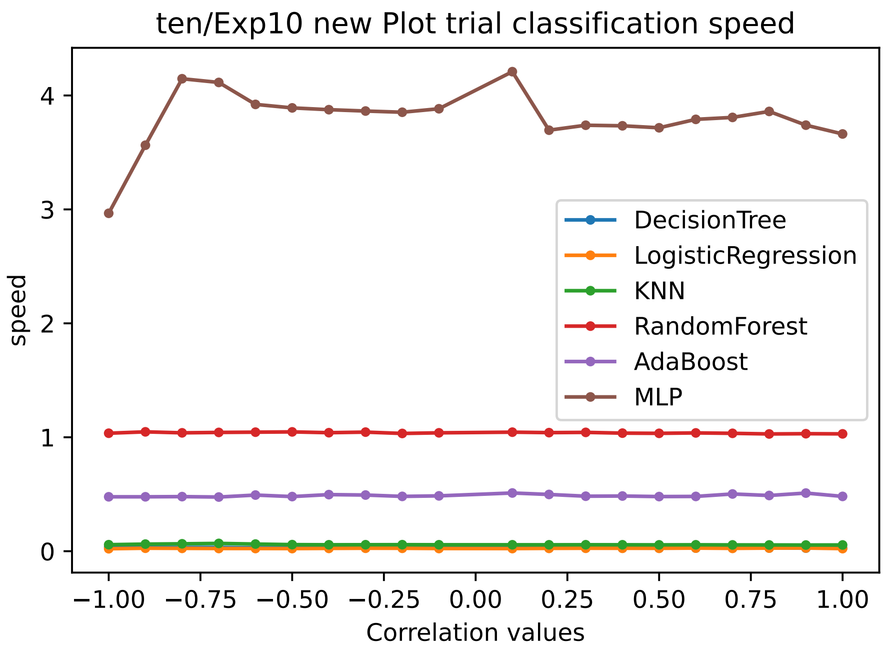

Figure 3.30: Exp10 speed of training

Correlation Values	DecisionTree	LogisticRegression	KNN	RandomForest	AdaBoost	MLP
-1	0.0363	0.0217	0.0572	1.0355	0.4775	2.9659
-0.9	0.0348	0.0262	0.0623	1.0477	0.4775	3.5634
-0.8	0.0330	0.0250	0.0649	1.0389	0.4792	4.1468
-0.7	0.0335	0.0238	0.0684	1.0426	0.4755	4.1143
-0.6	0.0300	0.0237	0.0628	1.0448	0.4929	3.9216
-0.5	0.0300	0.0231	0.0578	1.0473	0.4796	3.8910
-0.4	0.0282	0.0247	0.0564	1.0402	0.4968	3.8753
-0.3	0.0283	0.0260	0.0569	1.0452	0.4930	3.8634
-0.2	0.0283	0.0256	0.0573	1.0335	0.4811	3.8528
-0.1	0.0276	0.0236	0.0565	1.0390	0.4857	3.8834
0.1	0.0272	0.0232	0.0564	1.0448	0.5110	4.2088
0.2	0.0271	0.0250	0.0562	1.0407	0.4981	3.6953
0.3	0.0278	0.0259	0.0567	1.0430	0.4828	3.7388
0.4	0.0274	0.0257	0.0564	1.0363	0.4844	3.7338
0.5	0.0268	0.0256	0.0557	1.0346	0.4795	3.7164
0.6	0.0274	0.0272	0.0563	1.0383	0.4809	3.7903
0.7	0.0270	0.0249	0.0553	1.0346	0.5024	3.8075
0.8	0.0279	0.0273	0.0548	1.0291	0.4891	3.8599
0.9	0.0270	0.0280	0.0543	1.0313	0.5103	3.7394
1	0.0270	0.0228	0.0548	1.0299	0.4811	3.6621

Table 3.30: Exp 10 Speed

List of Figures

List of Tables