June-2023

Assignment 2 : AutoML

Machine Learning Course

Omer Yanai

Roei Zaady

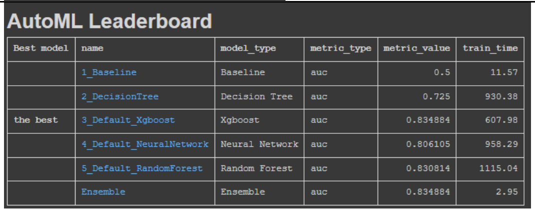
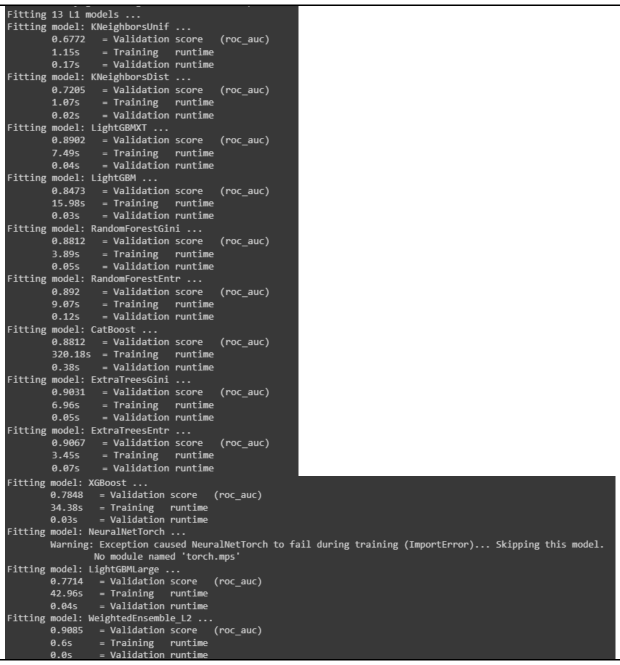
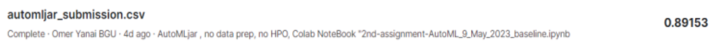
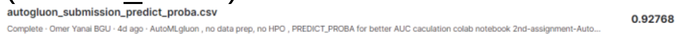
In this assignment we participated in a Kaggle competition, in which we dealt with a classification task, to predict if a drug is likely to damage the liver. The group's name on Kaggle is Omer & Roei.

Part A:

In this part, two AutoML frameworks were used, to see which algorithm can deliver the best AUC results on the competition's dataset.

The frameworks can be found in :
https://supervised.mljar.com/ and https://auto.gluon.ai/stable/index.html

Initial results

mljar Automated Machine Learning	AutoGluon
	
Best models : XGBoost and Ensemble - 0.834	Best model: "WeightedEnsemble_L2" - 0.9085 Second Best "ExtraTreesEnsemble" - 0.9067
Kaggle Leaderboard result : 0.901 (predict_Proba) 	Kaggle Leaderboard result : 0.927 (Predict_Proba) 

AutoML Leaderboard

Best model	name	model_type	metric_type	metric_value	train_time
	1_Baseline	Baseline	auc	0.5	11.57
	2_DecisionTree	Decision Tree	auc	0.725	930.38
the best	3_Default_Xgboost	Xgboost	auc	0.834884	607.98
	4_Default_NeuralNetwork	Neural Network	auc	0.806105	958.29
	5_Default_RandomForest	Random Forest	auc	0.830814	1115.04
	Ensemble	Ensemble	auc	0.834884	2.95

```
Fitting 13 L1 models ...
Fitting model: KNeighborsUnif ...
	0.6772	 = Validation score   (roc_auc)
	1.15s	 = Training   runtime
	0.17s	 = Validation runtime
Fitting model: KNeighborsDist ...
	0.7205	 = Validation score   (roc_auc)
	1.07s	 = Training   runtime
	0.02s	 = Validation runtime
Fitting model: LightGBMXT ...
	0.8902	 = Validation score   (roc_auc)
	7.49s	 = Training   runtime
	0.04s	 = Validation runtime
Fitting model: LightGBM ...
	0.8473	 = Validation score   (roc_auc)
	15.98s	 = Training   runtime
	0.03s	 = Validation runtime
Fitting model: RandomForestGini ...
	0.8812	 = Validation score   (roc_auc)
	3.89s	 = Training   runtime
	0.05s	 = Validation runtime
Fitting model: RandomForestEntr ...
	0.892	 = Validation score   (roc_auc)
	9.07s	 = Training   runtime
	0.12s	 = Validation runtime
Fitting model: CatBoost ...
	0.8812	 = Validation score   (roc_auc)
	320.18s	 = Training   runtime
	0.38s	 = Validation runtime
Fitting model: ExtraTreesGini ...
	0.9031	 = Validation score   (roc_auc)
	6.96s	 = Training   runtime
	0.05s	 = Validation runtime
Fitting model: ExtraTreesEntr ...
	0.9067	 = Validation score   (roc_auc)
	3.45s	 = Training   runtime
	0.07s	 = Validation runtime
Fitting model: XGBoost ...
	0.7848	 = Validation score   (roc_auc)
	34.38s	 = Training   runtime
	0.03s	 = Validation runtime
Fitting model: NeuralNetTorch ...
	Warning: Exception caused NeuralNetTorch to fail during training (ImportError)... Skipping this model.
		No module named 'torch.mps'
Fitting model: LightGBMLarge ...
	0.7714	 = Validation score   (roc_auc)
	42.96s	 = Training   runtime
	0.04s	 = Validation runtime
Fitting model: WeightedEnsemble_L2 ...
	0.9085	 = Validation score   (roc_auc)
	0.6s	 = Training   runtime
	0.0s	 = Validation runtime
```

automljar_submission.csv
Complete · Omer Yanai BGU · 4d ago · AutoMLjar , no data prep, no HPO, Colab NoteBook "2nd-assignment-AutoML_9_May_2023_baseline.ipynb
0.89153

autogluon_submission_predict_proba.csv
Complete · Omer Yanai BGU · 4d ago · AutoMLgluon , no data prep, no HPO , PREDICT_PROBA for better AUC caculation colab notebook 2nd-assignment-Auto...
0.92768

As can be seen, AutoGluon yielded higher AUC scores on the training set, than AutoMLjar. When applying the best models from each AutoML framework on the

Kaggle test set, the resulting AUC was higher than the training AUC (which is surprising).

Both frameworks found Ensemble based algorithms to be the top performing algorithms: XGBoost, Ensemble, WeightedEnseble_L2 and ExtraTreeEnsemble. These algorithms create an ensemble (group of models) that combines the predictions of multiple weak models (e.g. decision tree) to produce a more accurate prediction.

WeightedEnsemble_L2 is a weighted ensemble that returns a weighted average of predictions from individual models. Such models include KNN, ANN, SVMs and more [source].

Note
The initial experiments were done in a Kaggle Notebook environment, but due to its slow performance and low availability of TPU resources, we moved to Colab Notebooks instead.

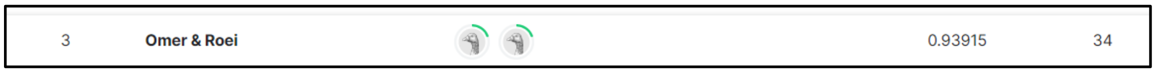
Part B:

Since the best estimator of Autogluon could not be fine-tuned, we selected the second best estimator, ExtraTreesClassifer, to focus on and optimize.

ExtraTrees is an ensemble method, similar to RandomForest, except it does not perform bagging by default and it randomly selects the values at which to split a feature on.

ExtraTrees is computationally efficient, which could help when searching for the right hyperparameters using GridSearch and RandomSearch.

Overall we submitted 34 attempt and reached best score of 0.93915



3	Omer & Roei	0.93915	34

Selected results (partial set of Kaggle submissions. Missing submissions had little changes in the model and on the AUC score).

Run	Method	Hyperparameters	Comment	AUC on train set	AUC on leaderboard
1	AutoMLjar - Default_Xgboost		binary output	0.8348	0.836
2	AutoMLjar - Default_Xgboost		predict_proba	0.8348	0.901
3	AutoMLgluon - WeightedEnsemble_L2		binary output	0.9085	0.859
4	AutoMLgluon - WeightedEnsemble_L2		predict_proba	0.9085	0.927
5	AutoMLgluon - WeightedEnsemble_L2		Impute missing by mean	0.9103	0.93121
6	AutoMLgluon - WeightedEnsemble_L2		Impute missing by median	0.9098	0.93121
7	ExtraTreesClassifier	n_estimators=250	impute missing by median	0.8867	0.93915
8	ExtraTreesClassifier	n_estimators=250, depth=5, max_feature=sqrt	impute missing by median	1	0.92768
9	ExtraTreesClassifier		Impute missing by median	0.9025	0.93915
10	ExtraTreesClassifier	depth=None	Impute missing by median	0.886	0.93386
11	ExtraTreesClassifier	n_estimators=250	Impute by median, remove useless features	1	0.93915
19	Catboost		Impute by median	0.8836	0.91005
20	ExtraTreesClassifier	n_estimators=250, depth=10, max_feature=sqrt	Impute by median	0.99	0.93298
22	ExtraTreesClassifier	n_estimators=400, depth=10, max_feature=sqrt	Impute by median	1	0.92945
26	ExtraTreesClassifier	n_estimators=250, depth=3, max_feature=sqrt	Impute by median	0.94	0.92592
27	ExtraTreesClassifier	n_estimators=300, max_depth=12, max_feature=30	Impute by median	0.8969	0.93474
34	ExtraTreesClassifier	trees=600, min split=6	only top 625 features	0.918	0.910

Data Processing

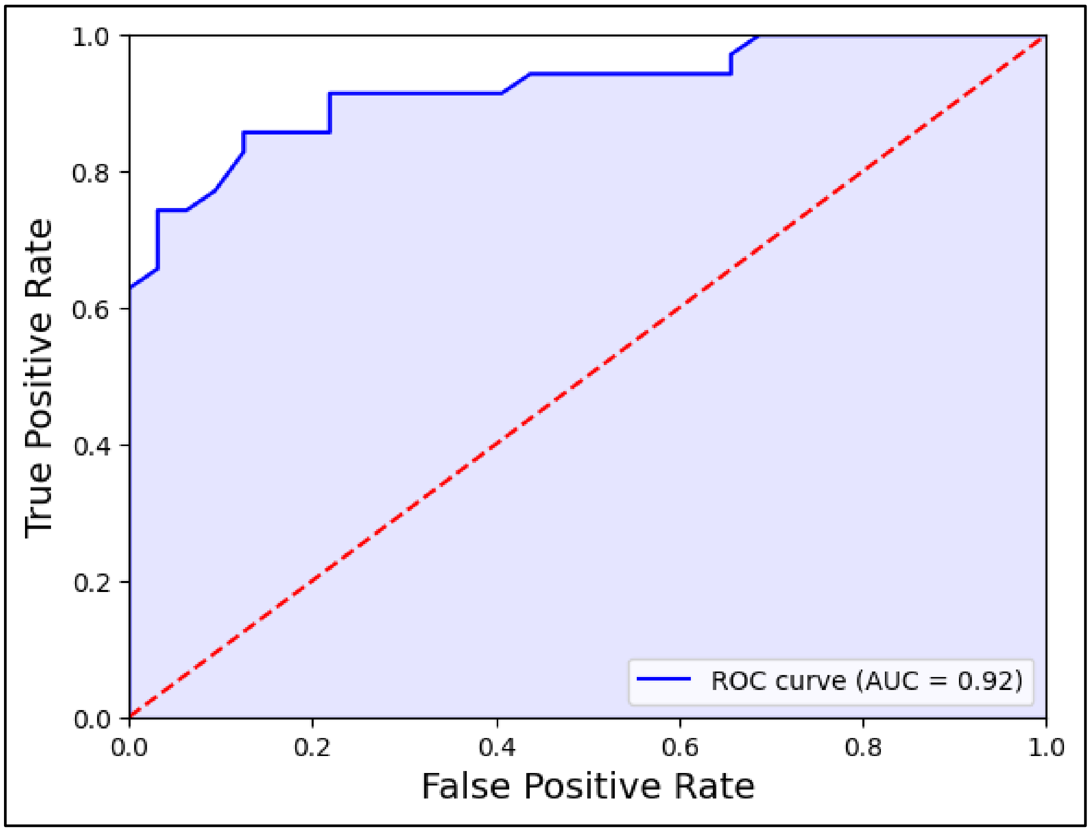
We experimented with various options for filling missing data using Mean and Median values. Until the twelfth attempt (and a few afterwards), all hyperparameters were chosen using a Grid Search algorithm with a 5 K-fold validation, to find the optimal hyperparameters values. In addition, we attempted to test the effects of removing unindicative columns as labeled as such by AutoGluon, but since they were unindicative and the trees depth wasn't limited, even if the ExtraTreesClassifier selected to split on them, it had no effect on the algorithm's performance and therefore, it did not change the AUC score.

Part D:

The notebook file can be found in the submission folder and at the link.

Part E:

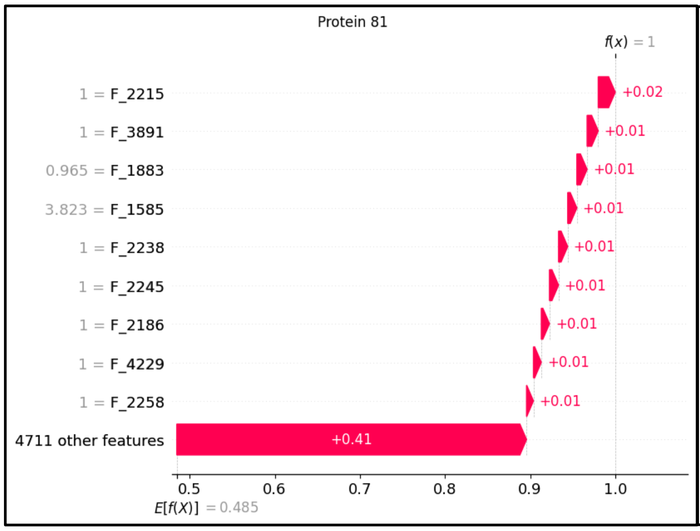
AUC Graph

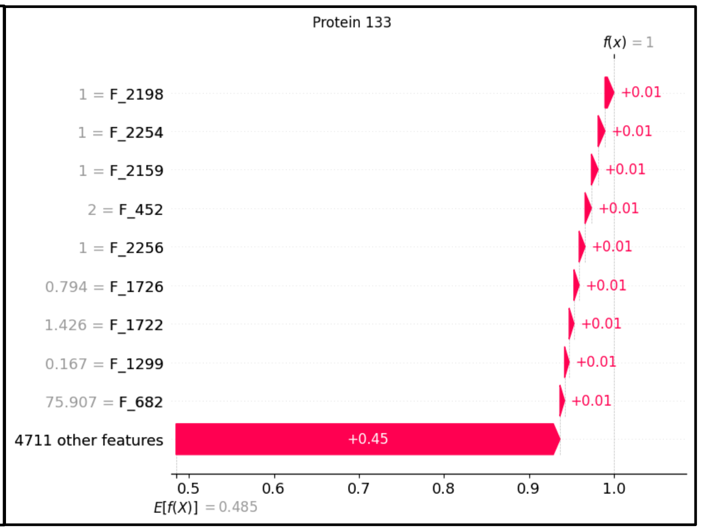


Since we do not have the test set true labels, we evaluated the mode AUC using the training set. We generated this graph by training the model on 80% of the samples and using the validation set of 20% of the samples for calculating the AUC.

Part F: SHAP analysis

Shap analysis explains the predictions of machine learning models. It assigns a value to each feature or input variable in a prediction, indicating its contribution to the prediction outcome.

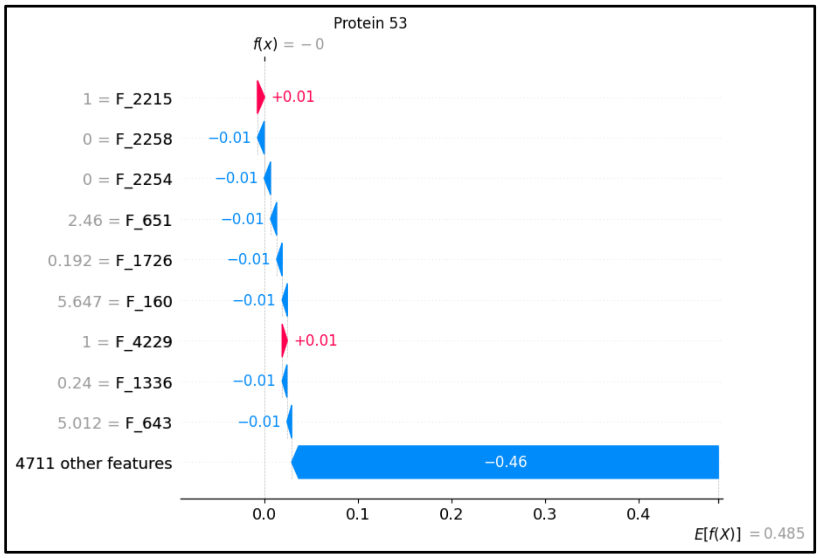




The Waterfall plots of drugs 81 and 133 show that the model strongly predicts these drugs as being in the positive class, i.e. damaging to the liver, based on the examples the model learned. The predictions in favor of the positive class are dictated by both the features that most heavily influenced the score, as well as by the sum of the other features' scores.

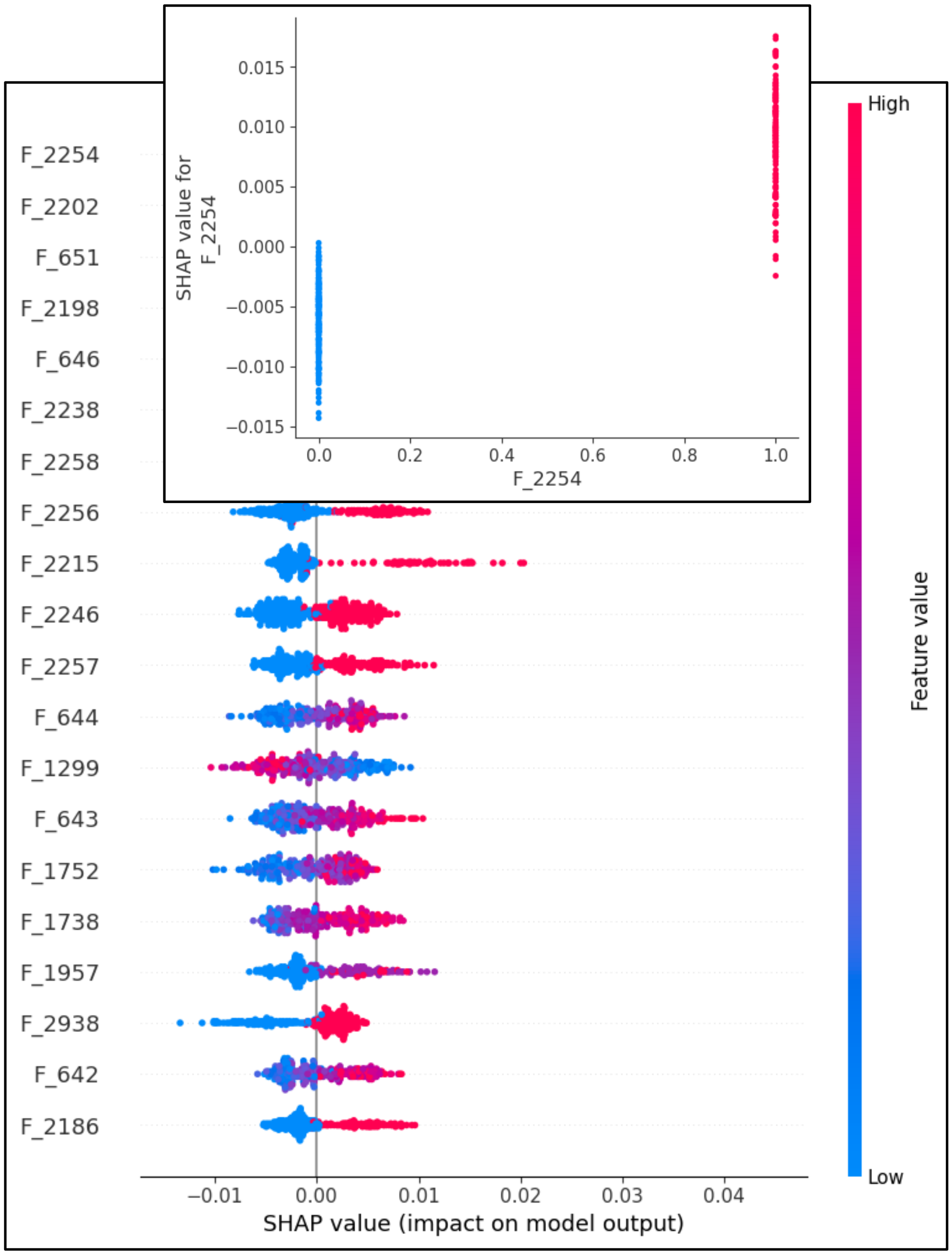
The Waterfall plot of sample 53 shows that despite having the most influential feature as well as the seventh most influential feature pointing to the positive class, overall, the sum of all other features point to the negative class, leading to a negative prediction, i.e. that this drug is not likely to damage the liver.

Part G:



<u>Feature Importance</u>

When analyzing feature importance we found that there were no dominant features with relative importance of more than 0.045. In addition, most of the dominant features have a clear divide, in which higher feature values increase the probability prediction that the drug is damaging to the liver, with a small minority of features with a more ambiguous divide on whether a higher feature value corresponds to a higher predicted probability, that a drug is damaging to the liver.

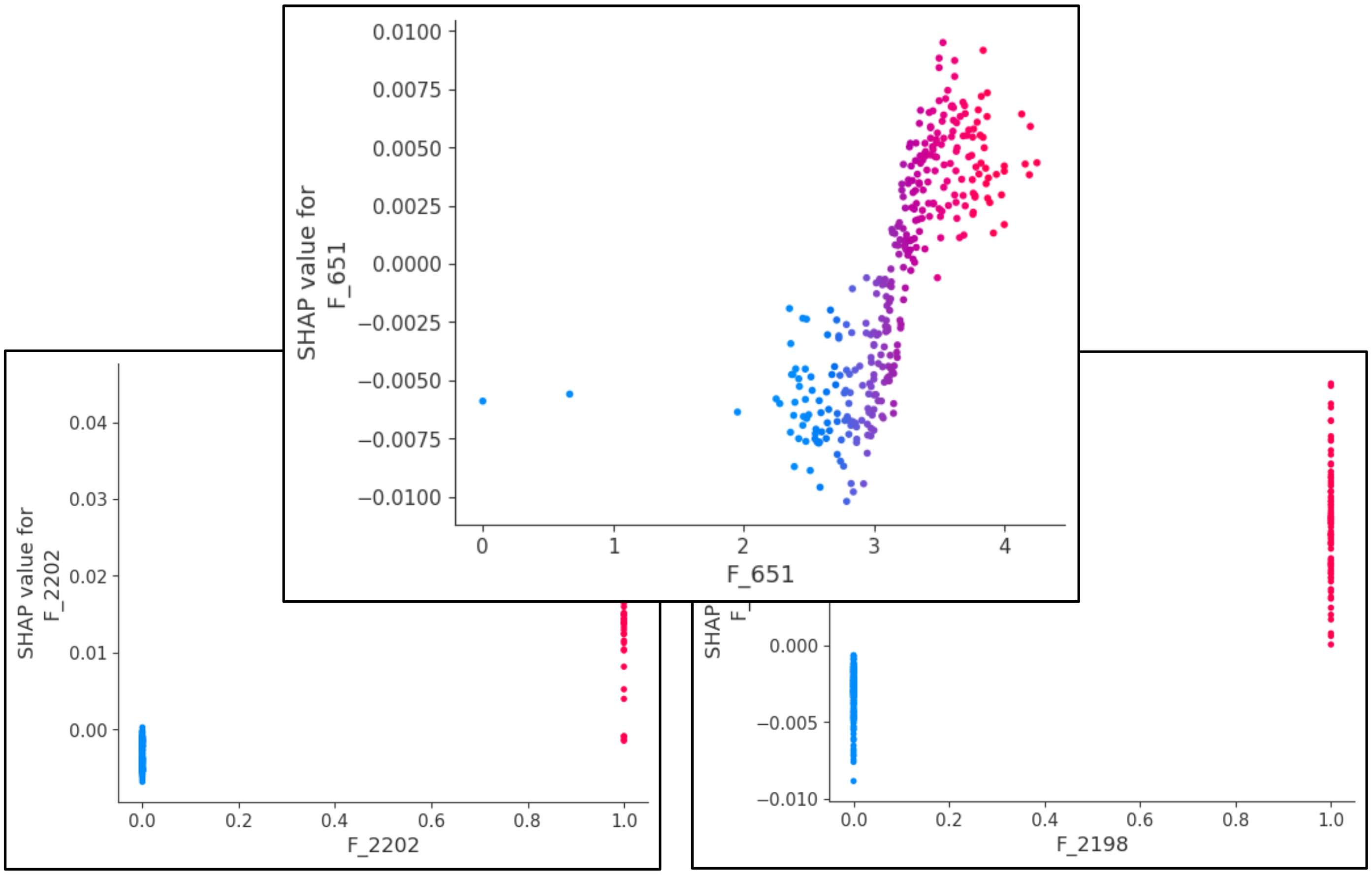


SHAP Dependence Contribution Plots:

Feature F_2254 is a binary feature, with a clear positive contribution toward the positive class (i.e. drug damages the liver) when the feature is positive (value of 1), and a negative contribution towards the negative class (i.e. drug does not damages the liver) when the feature is false (value of 0). There are a few outliers where this is the opposite case, but they are very few and therefore, negligible.

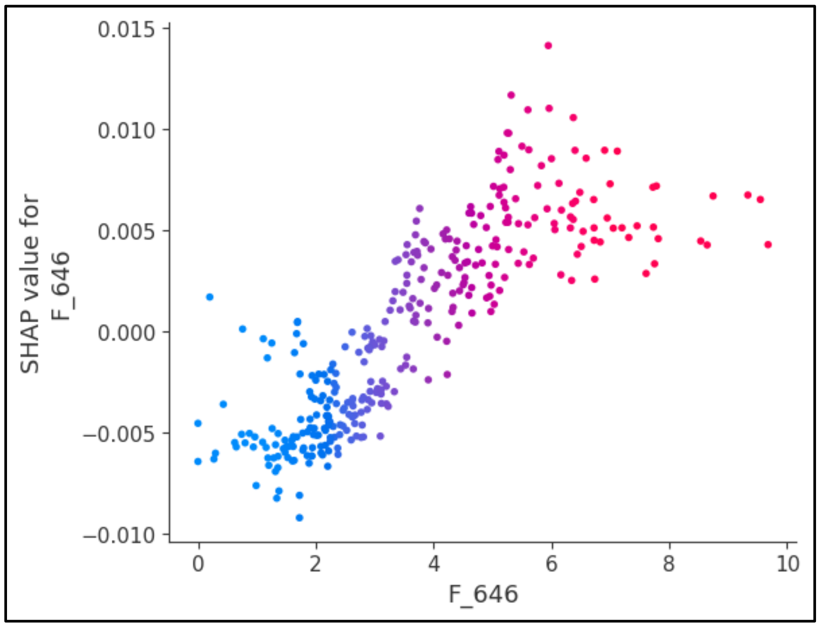
The same can be said for features “F_2022” and “F_2198”. However, where in “F_2254” the median distribution of the points seems to be around 0.01 and -0.006 for the positive and the negative classes respectively, for “F_2022” and “F_2198” the effect of the negative class on the prediction seems to be much lower than the the effect of the positive class, at around less than -0.01 and -0.03 for the negative class, and 0.008 and 0.018 for the positive class, for features “F_2022” and “F_2198” respectively. In “F_2202”, the negative class seems to contain a cleary larger amount of observations, than the positive class.

Features “F_651” and “F_646” have continuous values.



In the feature values of “F_651”, there seems to be less variance than in “F_646”. With most values being around the average of slightly above three and the great majority of total values located between two and four .

Values below 3 leading to a negative SHAP value, and values above 3 leading to a positive SHAP value. With that said, there is a lot of variance in the SHAP score for observations with a similar feature value, which suggests a strong influence of other features, on the overall score of the observations in relation to feature “F_651”.



As for feature “F_646”, it can be seen that all possible feature values are positive (>=0), and that the majority of values below 3.5 get a negative SHAP value, and above 3.5, a mostly positive value. In addition, 3 distinct clusters of values can be seen around 2, 4 and 6, having negative, small positive and higher positive SHAP values, respectively. Here as well, similar values can have different SHAP values within their cluster, suggesting that other features are influencing the overall score of observations, in relation to this feature.

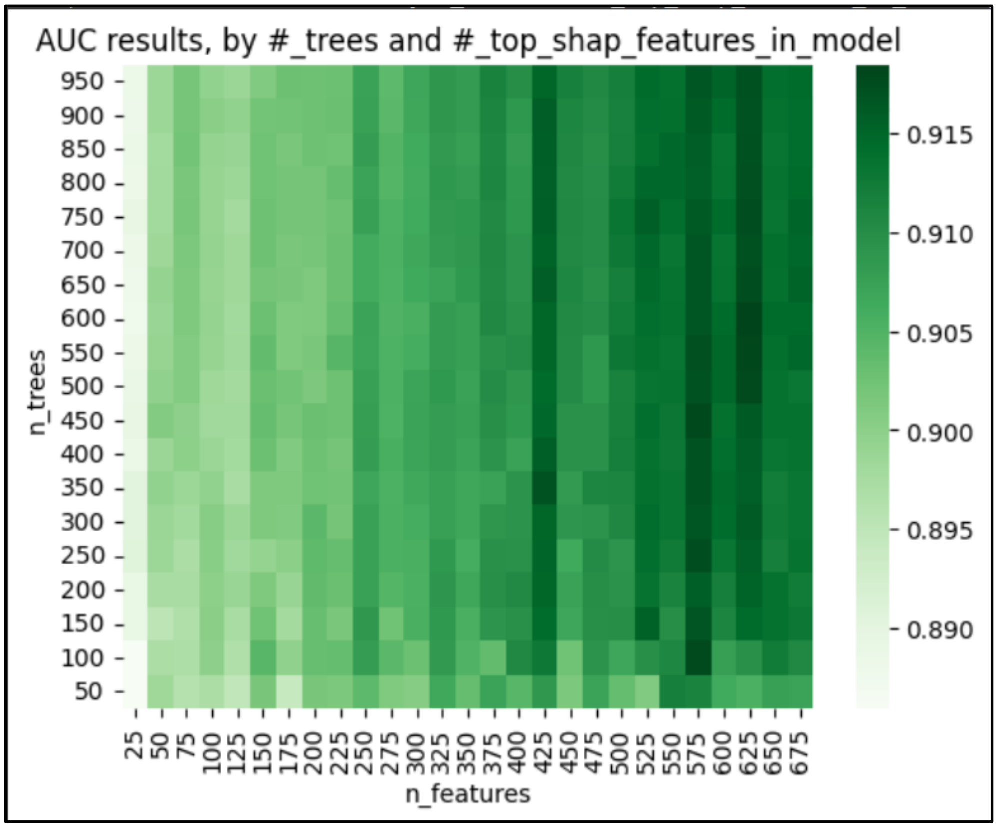
Appendix: Additional Analysis

After evaluating the features importance using SHAP analysis, we tried using the outcomes in order to perform Feature Selection. We evaluated the impact of building an ensemble model using only the top N features (based on their SHAP importance).

We experimented with multiple combinations of N_estimators (number of trees) and provided the model a limited set of features. The classifier used the default setting of max_features="sqrt" as the number of features to consider when looking for the best split.

We can see that the model performed better when more features were available and more trees were used.

The best result with 600 trees, 625 features yielded AUC of 0.9184 on the train set, but only 0.91 on the test set.



In summary, we were hoping that by limiting the model to selected features, we can improve give more weight to important features, improving its ability to generalize, but we could not show an improvement of performance using this method.