DECORATE Algorithm

Students:

Yuval Amit

Rotem Tragash

In appendix section you can see description about all py files and their outputs (csv and png files).

All files from our algorithm running exist in GitHub

1. **Algorithm Name:**

   DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples)

2. **Reference:**

   Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. Information Fusion 6(1): 99–111, 2004.

3. **Motivation for the algorithm (or which problems it tries to solve?)**

   "Constructing a diverse committee in which each hypothesis is as different as possible, while still maintaining consistency with the training data. All successful ensemble methods encourage diversity to some extent, **few have focused directly on the goal of maximizing diversity**. Existing methods that focus on achieving **diversity** are fairly **complex** and are not general metalearners like Bagging and Boosting which can be applied to any base learner to produce an effective.

   Methods such as Boosting, Bagging and Random Forests provide diversity by sub-sampling or re-weighting the existing training examples. If the training set is **small**, this **limits** the amount of ensemble diversity that these methods can obtain"

4. **Short Description:**

   DECORATE algorithm uses "strong" learners for building diverse committee. This algorithm combines ensemble and addition of different randomly artificial instances to training data set when building new committee learners.

   The labels of the new artificial examples disagree with the current committee decision, therefore diversity increases for the new trained learner.

"Decorate ensures diversity on an arbitrarily large set of additional artificial examples"

5. Pseudo-Code

DECORATE – Building the ensemble

Input: S – a labeled training set $S = (<x_1, y_1>, \ldots, <x_m, y_m>)$

E – Ensemble <classes[], C[]>

C_{iter} - leaner

C_size - Amount of learners in the ensemble

Imax - Maximum amount of iterations needed for building an ensemble

R_size - Factor to determine number of artificial examples to generate

1. For k=1 to CV_FOLD_NUM
   - 1.1. T – define train data set part of S
   - 1.2. $trial \leftarrow 1$
   - 1.3. $i \leftarrow 1$
   - 1.4. C_{iter} = DecisionTreeClassifier(T - Train C on fold train data set
   - 1.5. $E = E \cup C_{iter}$ - Add C_{iter} to E
   - 1.6. $Error = \frac{\sum_{TestSamples} I(y_{pred}, y_{true})}{\# TestSamples}, I(y_{pred}, y_{true}) = \begin{cases} 1 & y_{pred} \mathrel{!}= y_{true} \\ 0 & otherwise \end{cases}$
   - 1.7. While trial<Imax AND i<C_{iter}
     - 1.7.1.R = Artificial Examples with empty label (size = R_size*(S/10))
     - 1.7.2.R_label = Generate Label for artificial examples
     - 1.7.3.$T = T \cup R$
     - 1.7.4.C_{iter} = DecisionTreeClassifier(T) - Train C on fold train data set
     - 1.7.5.$E = E \cup C_{iter}$ - Add C_{iter} to E
     - 1.7.6.$Error_{new} = \frac{\sum_{TestSamples} I(y_{pred}, y_{true})}{\# TestSamples}, I(y_{pred}, y_{true}) = \begin{cases} 1 & y_{pred} \mathrel{!}= y_{true} \\ 0 & otherwise \end{cases}$
     - 1.7.7.if $Error_{new} < Error$
       - 1.7.7.1. $Error < Error_{new}$
         - 1.7.7.1.1. $i++$
     - 1.7.8.Else
       - 1.7.8.1. $E = E - C_{iter}$
       - 1.7.8.2. $trials++$
   - 1.8. END Whie

DECORATE – Classify an instance:

Input: S – Test instances needed to be labeled

$$T_{predict} = \mathrm{argmax}_{EnsembleClasses}\left(\frac{\sum_{c \in E} Predict_C(T)}{length(E)}\right), Predict_c - is\ predict\ probabilities\ of\ DecisionTreeClassifier$$

Figure 1:DECORATE Alogirthm

DECORATE – Create Artificial Examples:

Input: S – Train fold instances

R_size

For feature in S.features

if feature is binary (for categorical features: they are converted to one hot vectors)

Generate n (=R_size*|S|) values based on distribution derived from probability of the distinct values' occurrence.

Else (numerical features)

Generate n (=R_size*|S|) values based on Gaussian distribution defined by mean and standard deviation of the origin numerical feature values.

End For

DECORATE – Classify Artificial Examples:

Input: S – Train fold instances

$$P_s = Predict_{Ensemble}(S), generate\ probabilities\ for\ each\ class$$

$$P_s = \frac{P_s}{norm(P_s)}$$

$$Label = \operatorname{argmax}_{EnsembleClasses}\left(\frac{\frac{1}{P_s}}{sum(\frac{1}{p_s})}\right)$$

Figure 2: DECORATE Algorithm - Artificial Examples Generation

6. Algorithm Explanation:

There are 4 steps when running the algorithm:

- PreProcessing
- Hyper Parameters Random Search
- Algorithm (DECORATE)
- Evaluation

PreProcessing:

In pre-processing there is conversion of categorical field into one hot vectors which are concatenated into the origin dataframe.

Hyper Parameters Random Search:

50 iterations of 3 cross validation were made in order to find the hyper parameters (C_Size
, Imax, R_Size) which got the highest accuracy.

Algorithm (DECORATE):

The algorithm is depicted in Figure1 and Figure2.

CV_FOLD_NUM=10

Evaluation:

Calculated the following measurements for each fold in the 10 cv:

- Accuracy
- TPR
- FPR
- Prediction
- AUC
- PR-Curve
- Train Runtime
- Test Runtime

7. Illustration

- Table 5 below includes the initial baseball dataset we use for the running example
- Input for DECORATE algorithm:
  k of cv=2
  Imax=3
  C_size=3
  R_size=0.2
- You can see below the test results for the running example.

There are 2 folds, for each fold we run the DECORATE algorithm, each algorithm running makes fit for the first time (section 1.4 in pseudo code) with all train data set without artificial example and for the second and third time (section 1.7 in pseudo code) it makes fit with artificial examples.

Fold 1:

		y_predict		
		0	**1**	**2**
y_true	**0**	43	0	0
	1	0	2	0
	2	4	2	0

Table 1:First and second (with artificial examples) train Error=0.117

		y_predict		
		0	**1**	**2**
y_true	**0**	43	0	0
	1	1	1	0
	2	6	0	0

Table 2: Third train (with artificial examples) Error=0.137

Final fold1 accuracy = 0.882

Fold 2:

		y_predict		
		0	**1**	**2**
y_true	**0**	43	0	3
	1	2	2	0
	2	0	0	0

Table 3: First train Error=0.1

		y_predict		
		0	**1**	**2**
y_true	**0**	44	0	2
	1	2	2	0
	2	0	0	0

Table 4: Second (with artificial examples) and third trains (with artificial examples) Error=0.08

Final fold2 accuracy = 0.92

Final accuracy = 0.89

Hall_of_Fa	Position	Fielding_a	Slugging_p	On_base_	Batting_av	Strikeouts	Walks	RBIs	Home_run	Triples	Doubles	Hits	Runs	At_bats	Games_pl	Number_s
1	Outfield	0.98	0.555	0.377	0.305	1383	1402	2297	755	98	624	3771	2174	12364	3298	23
0	Second_ba	0.985	0.347	0.294	0.254	499	208	366	57	19	163	1022	378	4019	1165	13
0	Second_ba	0.974	0.353	0.343	0.286	223	453	394	9	48	249	1588	844	5557	1424	13
0	Third_base	0.955	0.368	0.34	0.269	447	414	303	37	49	188	1082	591	4019	1281	14
0	First_base	0.994	0.485	0.339	0.277	1059	594	1122	336	35	295	1832	823	6606	1959	17
0	Outfield	0.975	0.412	0.321	0.255	918	342	433	130	27	170	999	558	3912	1129	12
0	Shortstop	0.96	0.393	0.307	0.236	220	94	109	37	10	43	260	142	1104	568	10
0	Catcher	0.966	0.324	0.296	0.232	315	263	317	22	54	108	707	299	3048	1078	15
0	Second_ba	0.98	0.357	0.315	0.239	424	370	351	73	21	140	815	357	3404	1139	12
0	Outfield	0.981	0.41	0.336	0.3	310	223	501	47	45	255	1325	623	4418	1281	13
0	First_base	0.989	0.534	0.381	0.292	1556	894	1119	351	79	320	1848	1099	6332	1749	15
0	Shortstop	0.97	0.354	0.312	0.254	622	300	342	55	44	140	999	442	3927	1195	11
0	Outfield	0.975	0.471	0.36	0.255	1033	795	796	256	53	216	1281	811	5032	1541	13
0	Shortstop	0.956	0.343	0.307	0.254	636	250	296	36	25	138	846	390	3330	1236	15
0	Second_ba	0.977	0.288	0.291	0.245	482	302	282	13	19	126	1168	558	4760	1481	15
0	Outfield	0.979	0.433	0.33	0.286	706	423	852	206	49	359	2101	985	7339	2082	17
0	Outfield	0.968	0.353	0.307	0.28	267	138	377	32	26	170	1216	448	4345	1380	15
0	Outfield	0.979	0.381	0.346	0.307	377	311	427	31	50	236	1777	780	5789	1667	15
0	Second_ba	0.97	0.321	0.322	0.244	224	185	123	9	19	67	418	248	1715	643	10
0	Shortstop	0.967	0.292	0.31	0.234	280	227	156	8	13	75	505	211	2155	940	11
0	Shortstop	0.97	0.318	0.313	0.242	331	206	143	19	12	73	490	244	2026	873	10
0	Outfield	1	0.404	0.328	0.29	55	310	976	49	124	328	1841	870	6341	1635	14
2	First_base	0.974	0.446	0.395	0.329	294	952	1879	97	124	528	2995	1719	9101	2276	22
1	Shortstop	0.972	0.343	0.313	0.262	742	736	791	83	92	394	2677	1335	10230	2599	18
1	Shortstop	0.948	0.398	0.399	0.31	528	1302	1116	45	102	440	2749	1319	8856	2422	20
0	Catcher	0.971	0.334	0.288	0.249	241	124	296	17	33	106	660	246	2646	847	12
0	Outfield	0.981	0.453	0.29	0.252	1201	260	815	251	39	204	1302	614	5164	1432	14
2	Outfield	0.983	0.382	0.397	0.308	571	1198	586	29	109	317	2574	1322	8365	2189	15
0	Catcher	0.986	0.361	0.323	0.245	622	461	513	90	13	183	1010	397	4123	1370	17
0	Third_base	0.96	0.336	0.31	0.252	459	333	457	60	26	135	1103	386	4369	1324	13
0	Catcher	0.987	0.324	0.334	0.254	124	177	156	13	10	51	401	163	1579	544	10
0	Shortstop	0.96	0.286	0.287	0.22	198	127	86	9	5	56	309	167	1407	624	11
0	Third_base	0.933	0.314	0.326	0.246	363	592	390	13	76	174	1328	661	5388	1580	18
2	Outfield	0.97	0.534	0.395	0.318	518	774	1164	238	128	401	2019	1224	6353	1668	13
0	Second_ba	0.979	0.388	0.36	0.281	399	561	467	80	35	185	1296	725	4620	1300	11
0	Outfield	0.958	0.434	0.309	0.251	136	71	145	38	1	42	217	114	865	425	10
0	Catcher	0.992	0.344	0.307	0.252	344	207	304	50	9	94	712	201	2828	909	11
0	Catcher	0.986	0.429	0.358	0.256	577	545	540	155	15	128	915	432	3581	1212	14
0	Third_base	0.946	0.403	0.35	0.257	1126	852	773	189	43	234	1564	772	6082	1931	17
0	Outfield	0.98	0.325	0.312	0.264	164	187	222	9	23	107	775	339	2937	955	11
0	Third_base	0.971	0.297	0.36	0.251	165	382	196	1	13	76	573	285	2280	874	13
2	Third_base	0.943	0.442	0.363	0.307	182	473	987	96	103	315	1838	887	5984	1575	13
0	Outfield	0.985	0.432	0.351	0.278	926	762	1013	242	23	320	1981	964	7117	2039	19
2	Shortstop	0.944	0.358	0.355	0.279	487	827	591	32	77	320	2004	1048	7182	1913	16
0	Third_base	0.959	0.408	0.355	0.254	923	1031	1039	242	38	289	1790	982	7060	2019	16
1	First_base	0.994	0.5	0.333	0.274	1236	763	1636	512	90	407	2583	1305	9421	2528	19
0	Outfield	0.983	0.355	0.337	0.27	318	292	288	19	28	143	811	430	3007	972	12
0	Outfield	0.98	0.466	0.338	0.256	1234	551	716	241	30	216	1219	715	4759	1428	12
0	Outfield	0.954	0.359	0.379	0.291		440	255	16	47	83	962	580	3306	866	10
0	Second_ba	0.986	0.347	0.34	0.278	209	304	314	18	9	163	938	418	3378	941	10
0	Outfield	0.955	0.33	0.32	0.267		279	391	10	47	128	1073	516	4014	1100	10
0	Shortstop	0.935	0.303	0.321	0.243	142	396	429	10	38	142	1009	532	4146	1223	11
0	Shortstop	0.953	0.391	0.355	0.284	627	748	710	79	71	442	2165	1130	7629	2016	18
0	Catcher	0.987	0.287	0.288	0.226	168	87	66	9	3	31	237	54	1049	393	10
0	Catcher	0.982	0.35	0.273	0.23	610	172	375	81	18	123	765	250	3330	1017	10
0	Catcher	0.99	0.409	0.351	0.27	470	421	449	104	17	150	969	393	3586	1141	13
0	Catcher	0.983	0.391	0.33	0.269	163	143	220	26	11	95	432	163	1605	546	10
0	Outfield	0.982	0.439	0.347	0.277	638	521	703	164	57	229	1424	833	5145	1544	14
0	Outfield	0.98	0.389	0.342	0.29	258	258	272	25	51	165	1010	450	3477	1019	10
0	Designate	0.977	0.436	0.346	0.26	1069	805	1276	338	28	366	2135	1236	8198	2292	19
0	First_base	0.98	0.334	0.292	0.231	150	54	90	14	4	18	153	79	661	393	10
0	Outfield	0.956	0.393	0.362	0.311	14	425	617	39	82	182	1759	955	5660	1463	12
0	Second_ba	0.973	0.345	0.319	0.283	243	260	360	22	31	196	1473	685	5208	1320	11
2	First_base	0.981	0.435	0.361	0.308	270	616	1575	86	243	473	2930	1600	9526	2386	20
0	Shortstop	0.977	0.28	0.302	0.228	839	576	389	20	33	175	1316	676	5784	2016	18
0	Third_base	0.964	0.406	0.343	0.279	776	836	1106	201	56	425	2514	1151	8995	2405	18
0	Outfield	0.985	0.445	0.333	0.281	636	470	942	206	66	311	1823	865	6478	1741	15
1	Catcher	0.99	0.476	0.345	0.267	1278	891	1376	389	24	381	2048	1091	7658	2158	17
0	Catcher	0.99	0.299	0.322	0.242	251	328	260	18	6	98	696	214	2878	982	12
0	Catcher	0.988	0.317	0.295	0.255	45	62	108	0	12	46	287	83	1125	411	10
0	Outfield	0.977	0.379	0.329	0.274	551	349	476	79	30	190	1274	610	4651	1500	17
0	Catcher	0.942	0.387	0.34	0.256	572	478	533	55	67	203	978	549	3821	1062	15
0	Second_ba	0.968	0.355	0.316	0.249	268	284	387	36	23	167	755	334	3028	912	11
0	Catcher	0.986	0.299	0.278	0.243	117	78	206	6	6	71	441	150	1813	663	15
0	Catcher	0.972	0.201	0.194	0.17	81	88	193	2	21	45	516	138	3028	947	11
0	Outfield	0.974	0.522	0.359	0.3	694	435	898	242	59	299	1550	809	5163	1350	11
0	First_base	0.992	0.367	0.351	0.258	347	380	289	54	16	100	690	312	2679	1349	17
0	Second_ba	0.978	0.387	0.341	0.262	606	428	391	75	30	177	970	523	3700	1071	10
0	Shortstop	0.959	0.344	0.297	0.236	422	210	278	49	9	109	603	236	2553	853	11
1	Catcher	0.989	0.482	0.35	0.285	414	704	1430	358	49	321	2150	1175	7555	2120	19
0	Catcher	0.989	0.255	0.26	0.216	134	76	78	3	3	37	287	96	1330	561	11
0	Outfield	0.989	0.344	0.309	0.255	569	298	343	58	23	150	1053	422	4136	1383	14
0	Catcher	0.982	0.374	0.322	0.267	196	160	256	23	29	88	539	196	2018	709	11
0	Third_base	0.949	0.336	0.306	0.244	252	142	171	27	10	60	425	204	1745	612	10
0	Outfield	0.96	0.351	0.353	0.258	451	619	345	28	74	190	1171	749	4536	1228	11
0	Third_base	0.938	0.327	0.309	0.236	329	221	275	27	11	90	499	214	2117	970	15
0	Second_ba	0.935	0.354	0.301	0.267	129	268	706	33	95	208	1521	819	5706	1383	13
0	Outfield	0.966	0.369	0.345	0.287	161	344	324	17	75	139	1205	629	4192	1147	11
0	Outfield	0.97	0.359	0.326	0.273	287	236	354	29	20	144	861	310	3151	1217	14
0	First_base	0.992	0.444	0.339	0.249	395	234	276	76	13	85	432	220	1738	600	10
0	Catcher	0.964	0.262	0.268	0.217	73	23	29	0	5	12	106	44	488	243	11
0	Second_ba	0.976	0.366	0.423	0.271	452	1153	379	41	35	236	1216	966	4494	1338	12
0	Outfield	0.963	0.46	0.395	0.301	310	331	340	50	51	133	726	467	2415	767	10
0	Outfield	0.988	0.382	0.305	0.25	877	449	620	134	55	282	1513	776	6042	1947	17
0	Second_ba	0.979	0.327	0.33	0.258	462	552	308	21	62	178	1366	731	5296	1444	12
0	Second_ba	0.975	0.358	0.292	0.248	407	202	451	62	20	160	874	347	3519	1002	11
0	First_base	0.989	0.401	0.402	0.287	436	1092	695	44	109	319	1696	1151	5904	1615	13
0	Third_base	0.957	0.356	0.352	0.272	515	723	848	43	67	276	1751	883	6440	1867	18
0	Catcher	0.984	0.317	0.269	0.219	246	96	148	26	5	56	320	117	1462	551	12
0	First_base	0.992	0.396	0.363	0.282	662	653	658	100	21	250	1478	643	5233	1538	12
0	Second_ba	0.982	0.366	0.315	0.254	558	462	556	106	40	221	1415	692	5562	1540	12

Table 5: baseball data set

8. Strengths

Decorate algorithm strength is focused on increasing the diversity of the examples. Allowing it to outperform other algorithms on small database, making better generalization on these datasets.

9. Drawbacks

The most significant drawback of DECORATE algorithm is that it is very time consuming, making it less desirable on large datasets.

10. Experimental Results

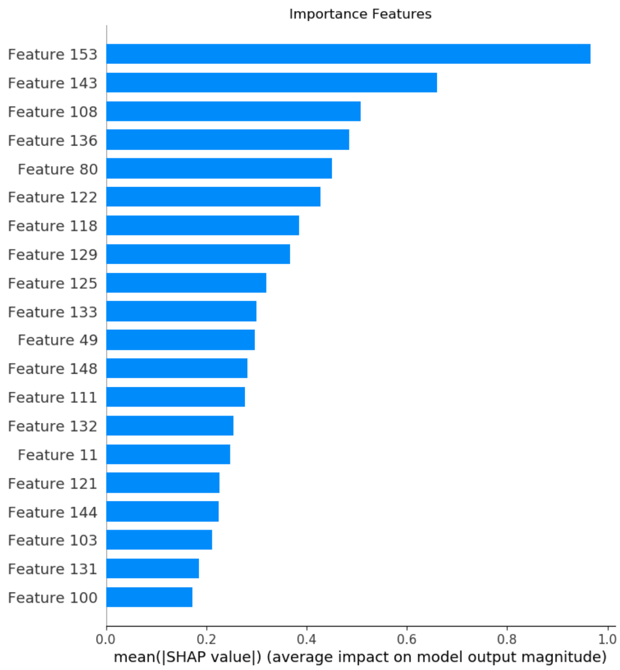
Meta learner scores:

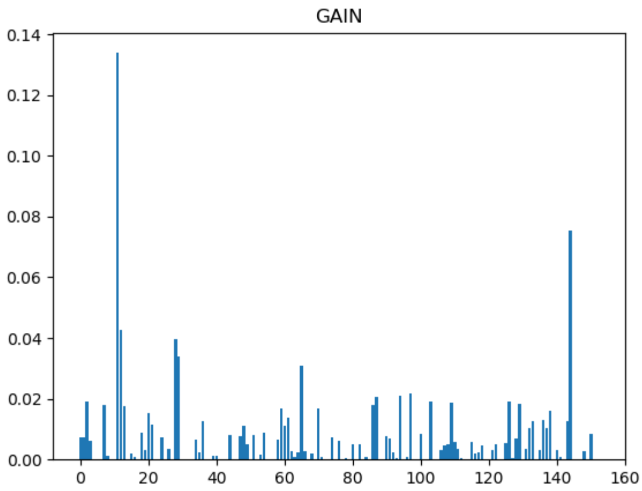
We tested the meta learner with Leave One Out on the datasets which leads to the following scores:

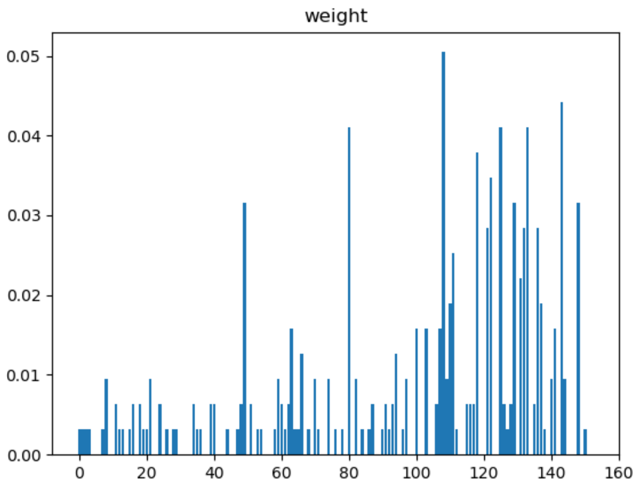
ACC	TPR	FPR	PPV	AUC ROC	AUC PRECISION
0.69	0.69	0.44	0.67	0.63	0.84

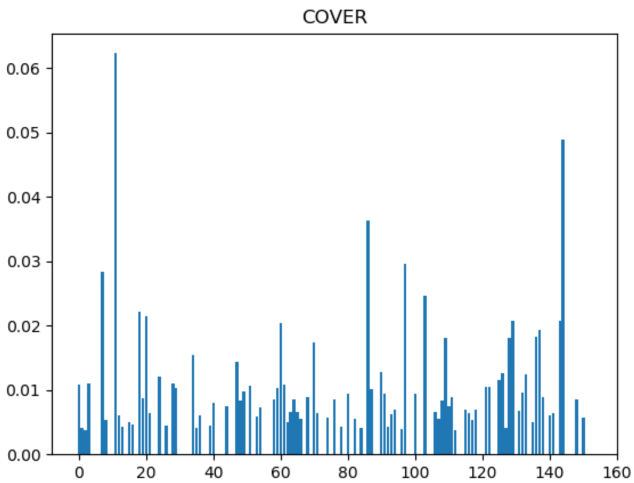
Feature importance graphs.

Shap




GAIN
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100
120
140
160


weight
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
120
140
160


COVER
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
120
140
160

11. Conclusions

The algorithms perform better or at least similarly in small datasets on all measures we checked. We would recommend using it on these sets. However, in case of large datasets, it should be avoided and other algorithms should be used.

12. Citations

- Diversity is known to be an important factor which affects the generalization performance of Ensemble classifiers. **(**Li X, Wang L, Sung E. AdaBoost with SVM based component classifier. Eng Appl Artif Intell. 2008;21(5):785–795**)**
- To improve the performance of the single classifier approaches the combination of multiple classifiers has been proposed in the field of machine learning. (Nanni, L., & Lumini, A. (2009). An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. Expert Systems with Applications, 36, 3028-3033)
- Melville and Mooney (2005) presented a new method RN/11/02 Page 4 Ensemble Learning Martin Sewell for generating ensembles, DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), that directly constructs diverse hypotheses using additional artificially-constructed training examples. Their approach consistently outperformed the base classifier, bagging and random forests; and outperformed AdaBoost on small training sets and achieved comparable performance on larger training sets.(G. Brown, J. Wyatt, R. Harris, and X. Yao, “Diversity creation methods: A survey and categorisation,” Inf. Fusion, vol. 6, no. 1, pp. 5–20, 2005.)

Appendix:

Files:

.py file	File description	files outputs
main	Includes final report framework generation and all function calls (decorate and gbm)	*tests_results.csv*- final report
pre_processing	This file include the preprocessing of the data set, such as converting categorical features into one hot vectors	
common	Includes the functions of measurements calculations	
gbm	GBM algorithm , it includes hyper parameters random search and 10 cv	
decorate	DECORATE algorithm , it includes hyper parameters random search and 10 cv	
statistic_test	Input for this file: *tests_results.csv* This file includes Mann-Whitney U test, for comparing DECORATE and GBM algorithms	*Statistic_test.txt* – includes the statistic tests results in the first row, and in the second row if there was a rejection of the test
meta_learner	Input for this file: *tests_results.csv* This file includes meta leaner which define which algorithm will be better according to meta features	• *meta_added_class.csv* - meta features file with the **class which is set according to tests_results.csv** • *shap.csv* - importance • *importance results.csv* - importance features according to **gain, weight and cover** measurements. • *y_predicted.csv* – **predicted good algorithm (=classes) for each data set file** • *meta_results.csv* – test measurements (Accuracy, TPR, FPR, PPV, AUC_roc, AUC_pr) • *importance_features.png* - importance features chart • *gain.png* - gain chart • *weight.png* - weight chart • *cover.png* - cover