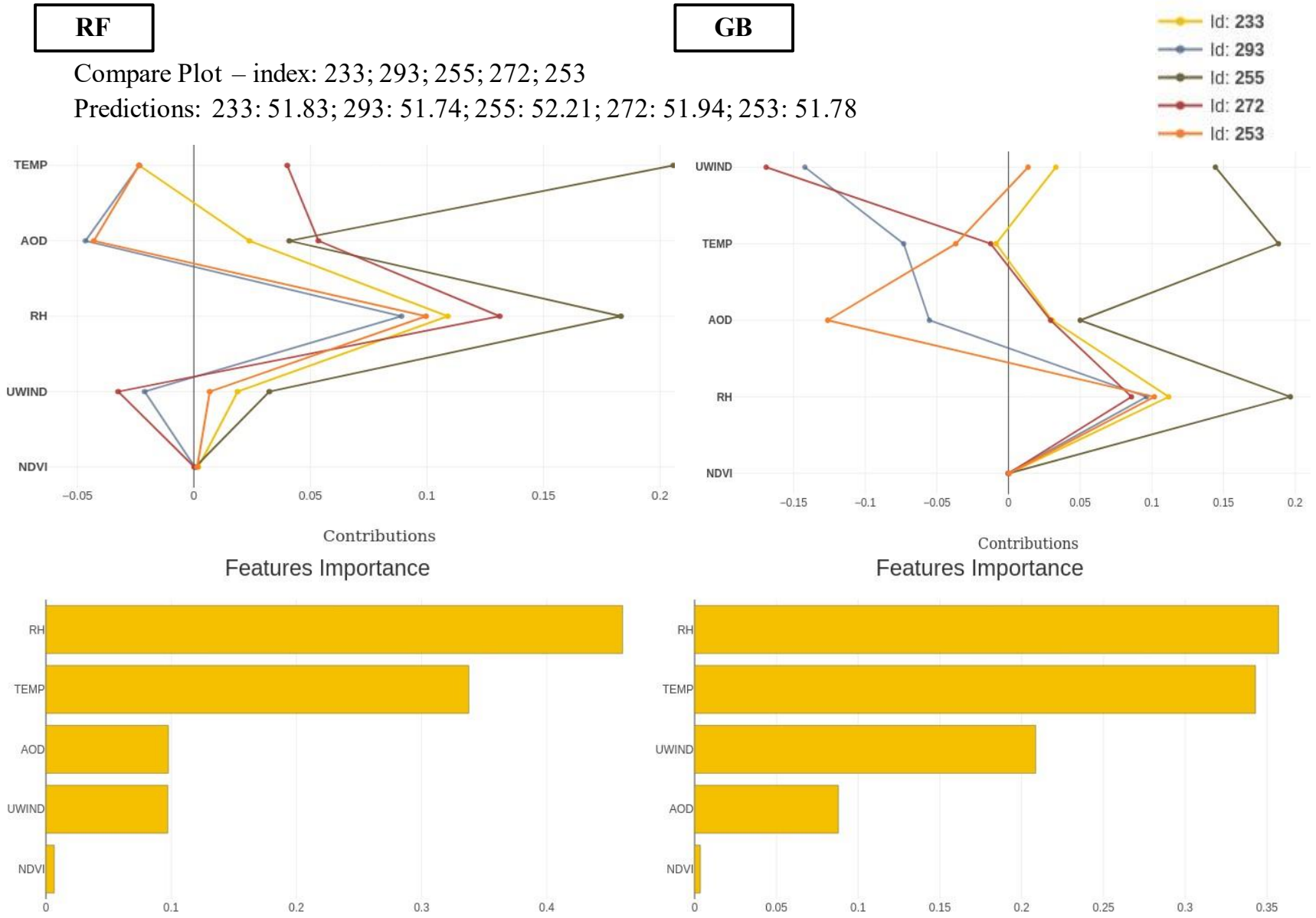

Figure S1: RF and GB local comparison line plots are shown at the top and overall contributions at the bottom for July 2019.

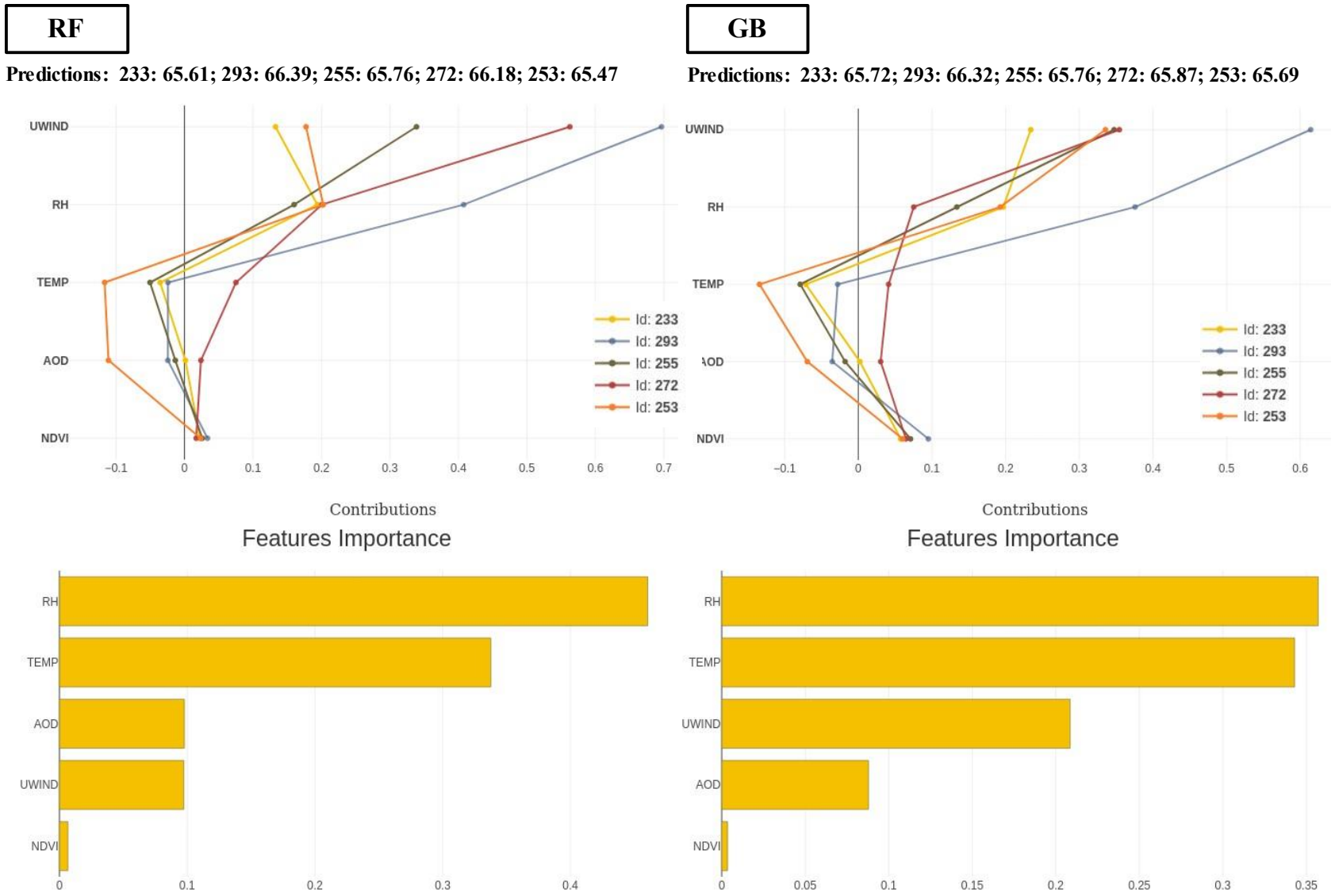

Figure S2: RF and GB local comparison line plots are shown at the top and overall contributions at the bottom for August 2019.

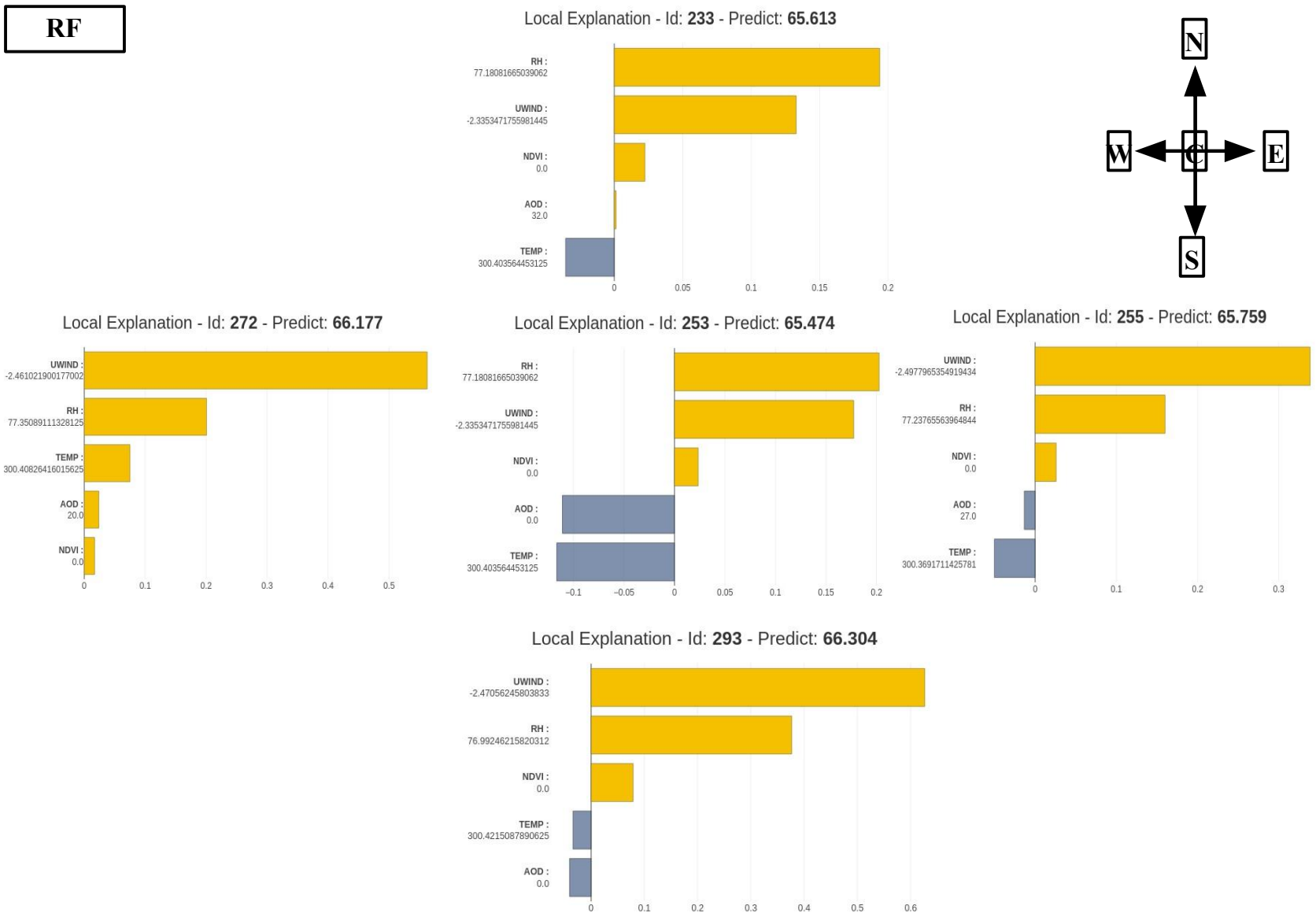

Figure S3: RF local comparison based on the direction for August 2019.

RF - AOD

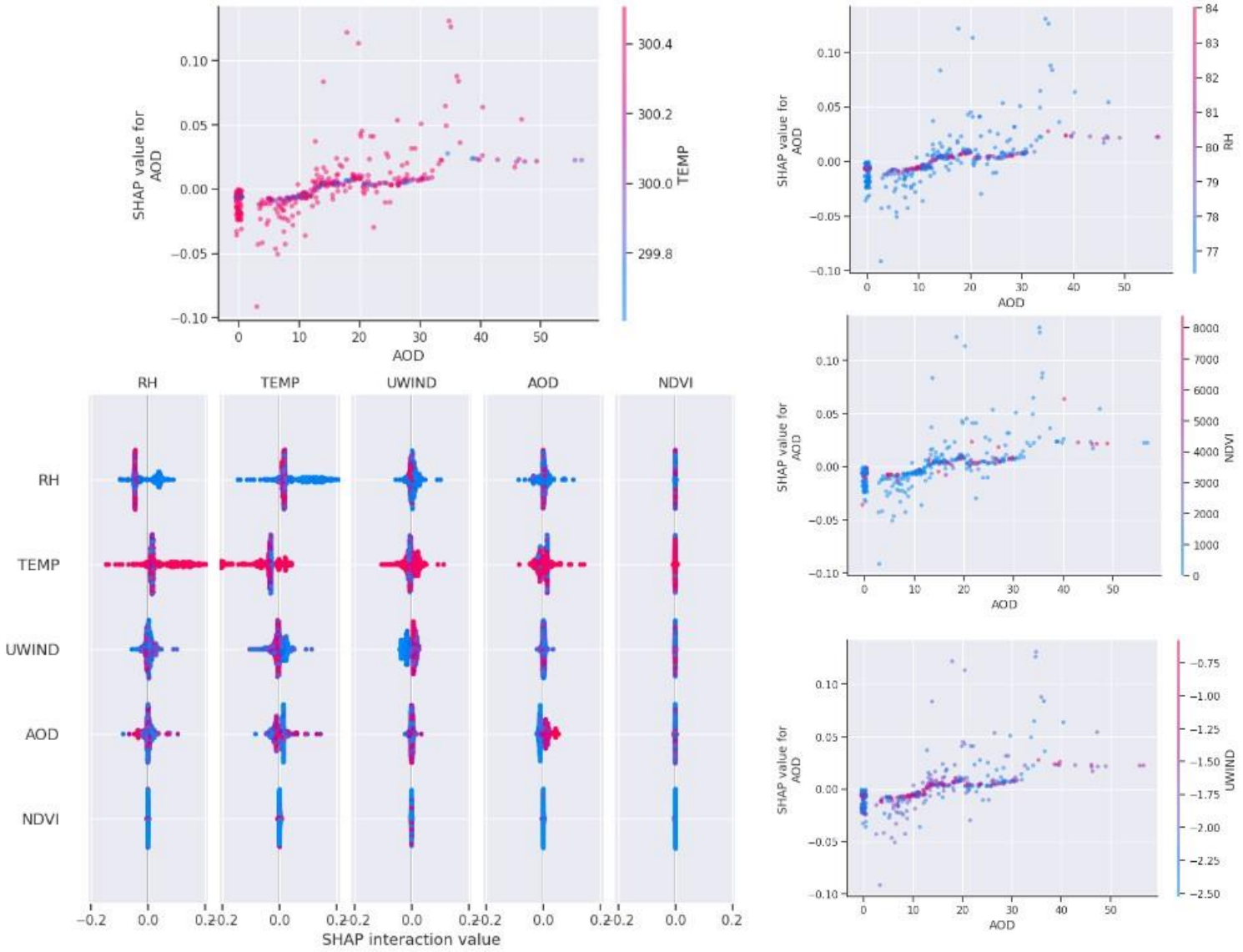

Figure S4: Showing the local interpretation comparison plot for the AOD feature in RF for July 2019.

GB - AOD

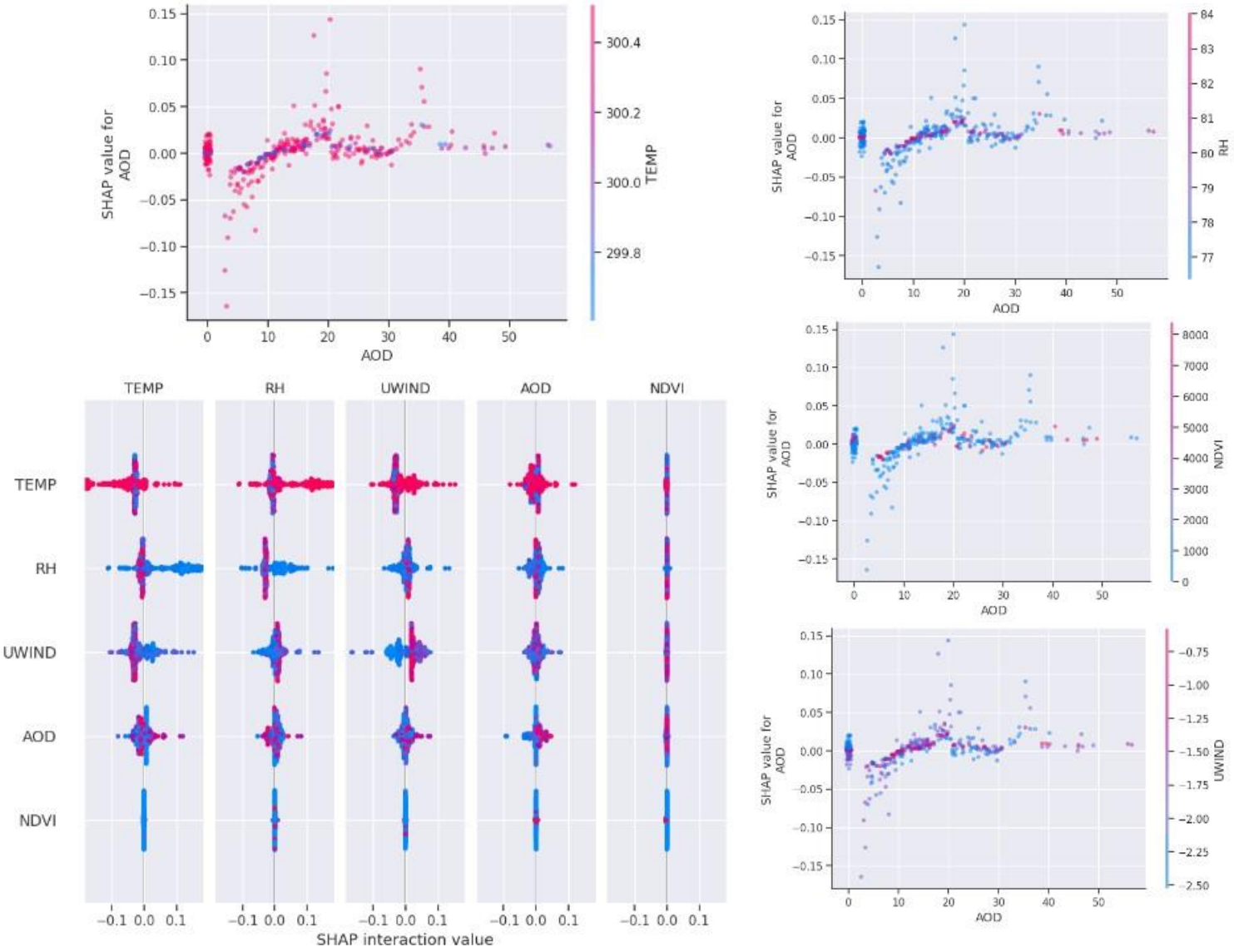

Figure S4 cont. Same as figure S4 but for the GB method.

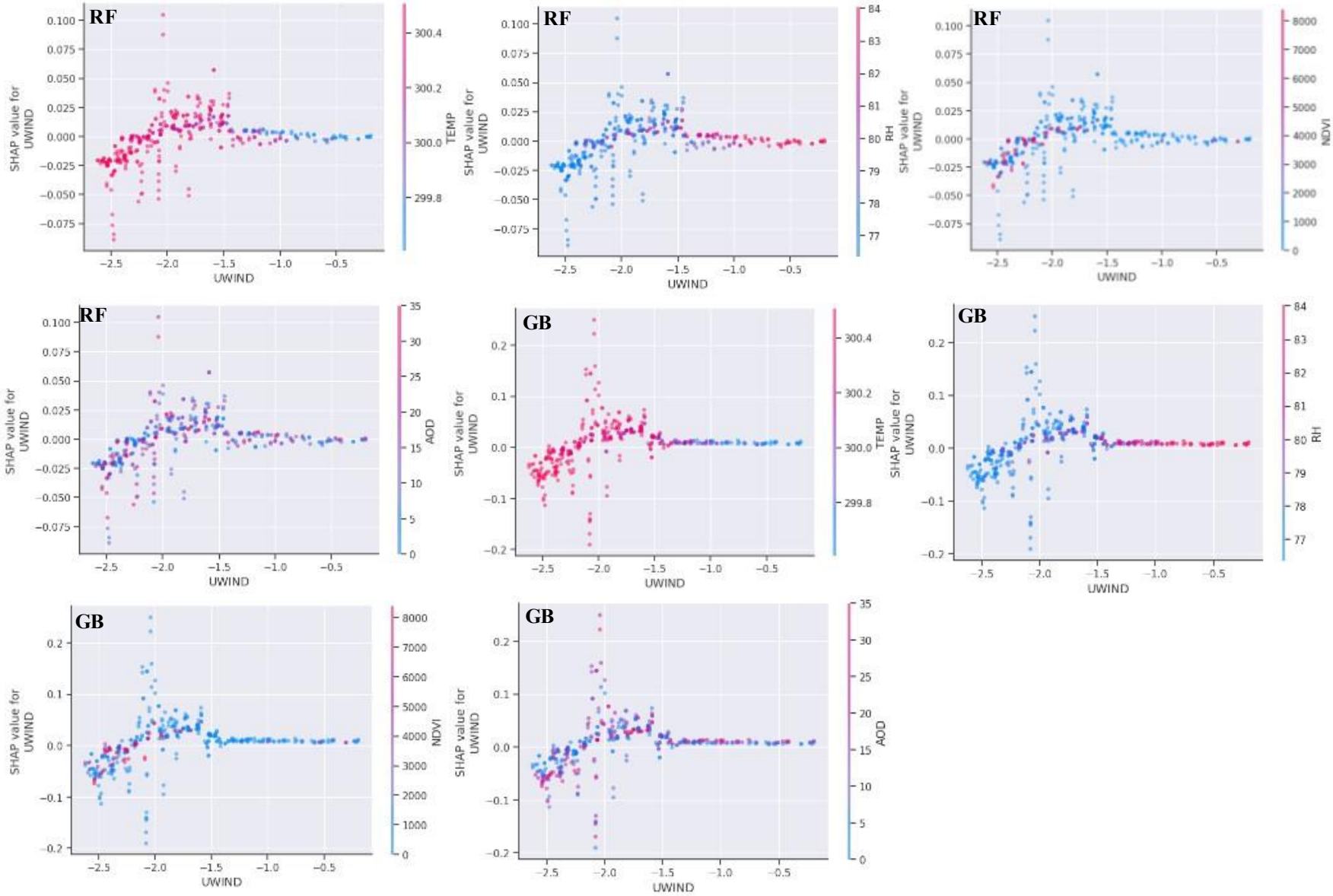

Figure S5: Showing the local interpretation comparison plot for the UWIND feature in RF and GB methods for July 2019.

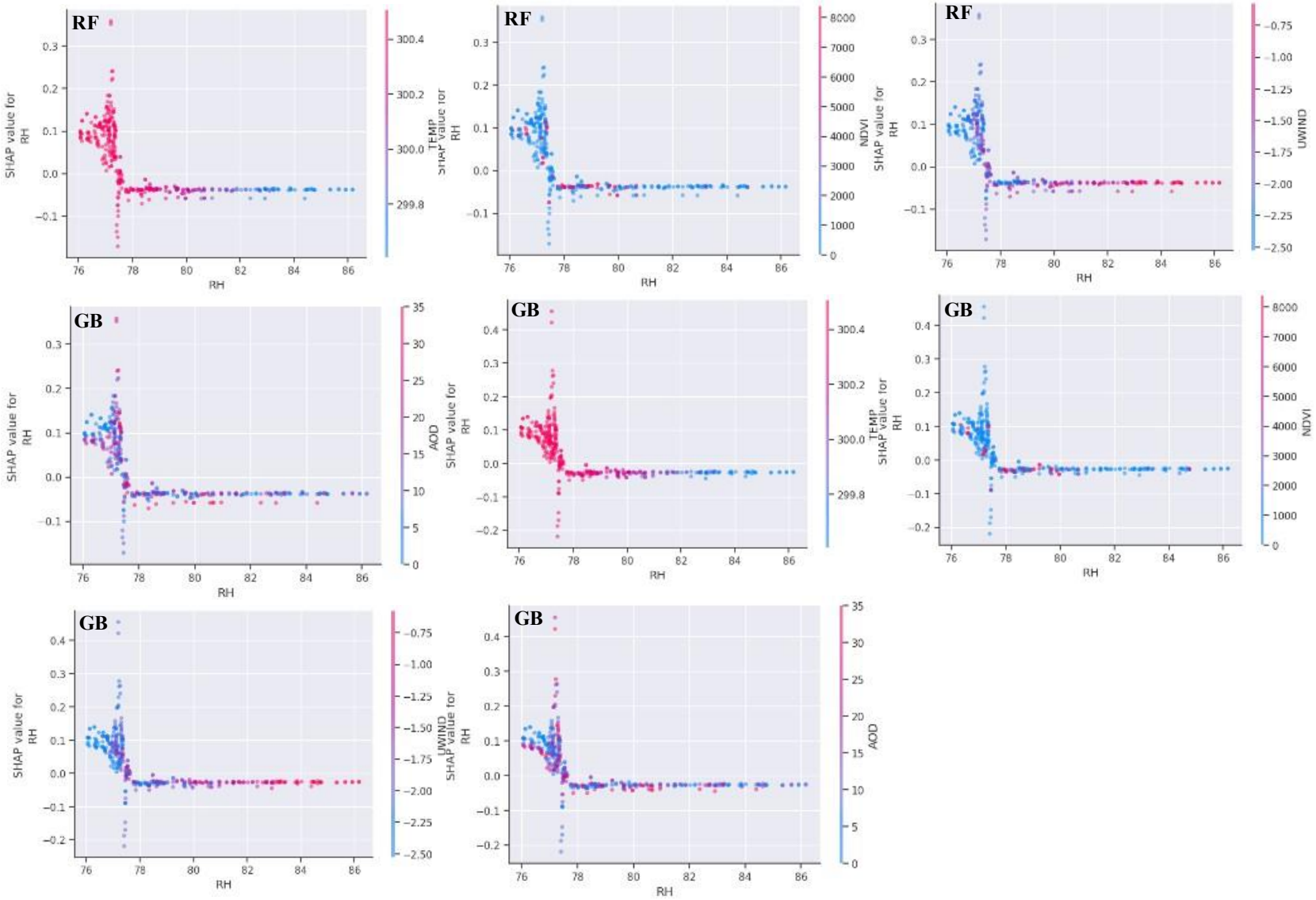

Figure S5 cont. Same as figure S5 but for the RH feature.

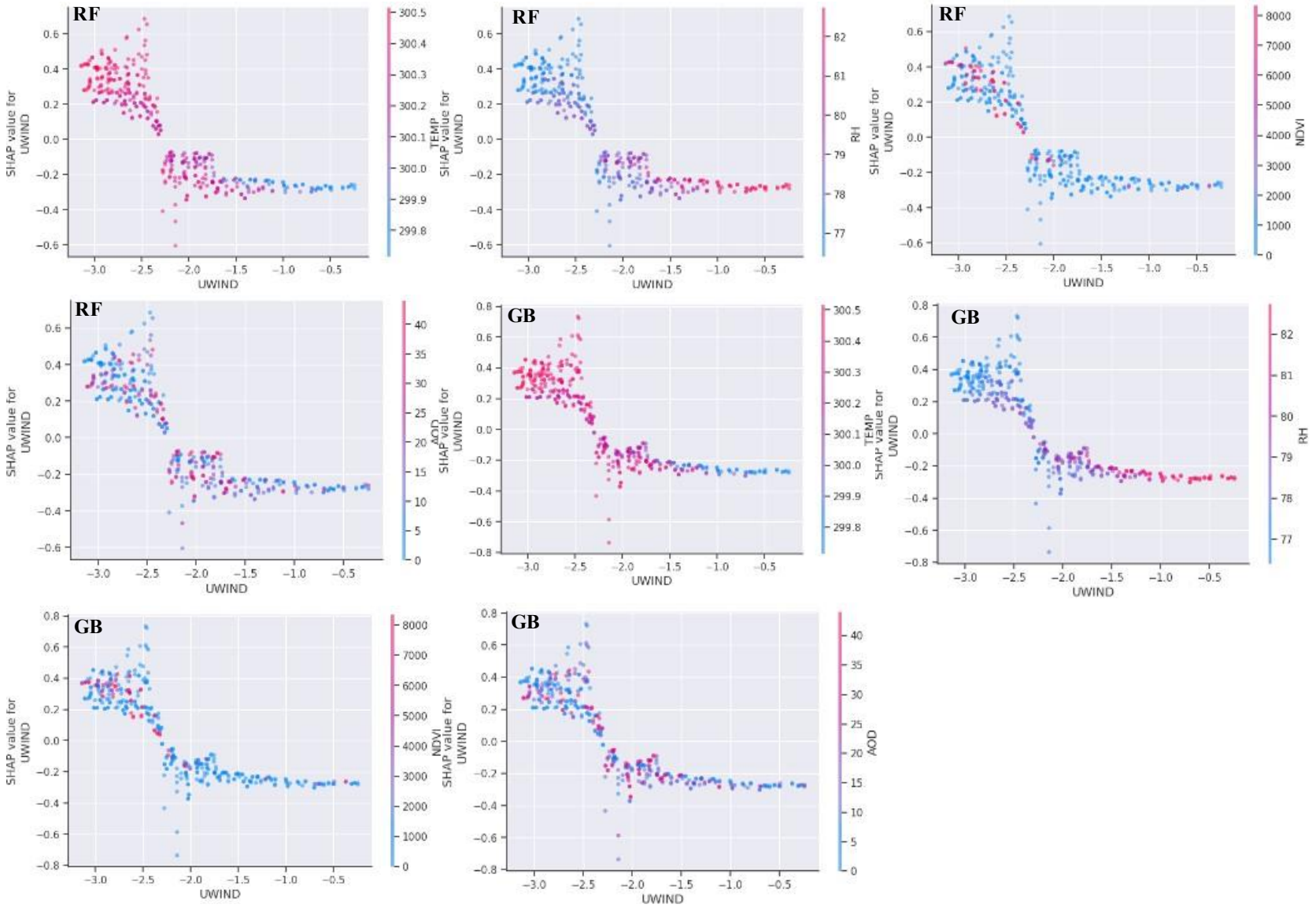

Figure S6: Showing the local interpretation comparison plot for the UWIND feature in RF and GB methods for August 2019.

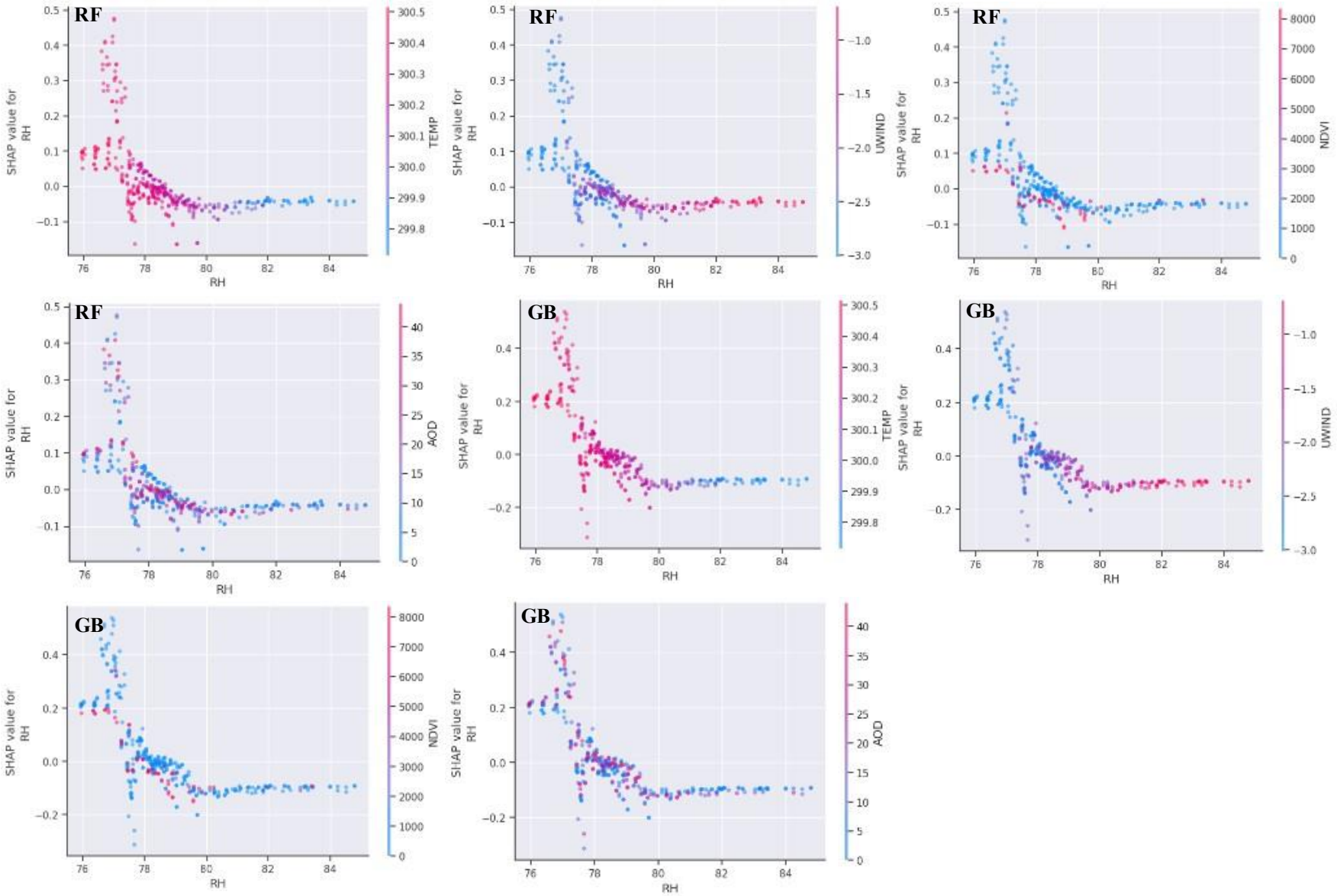

Figure S6 cont. Same as figure S6 but for the RH feature.

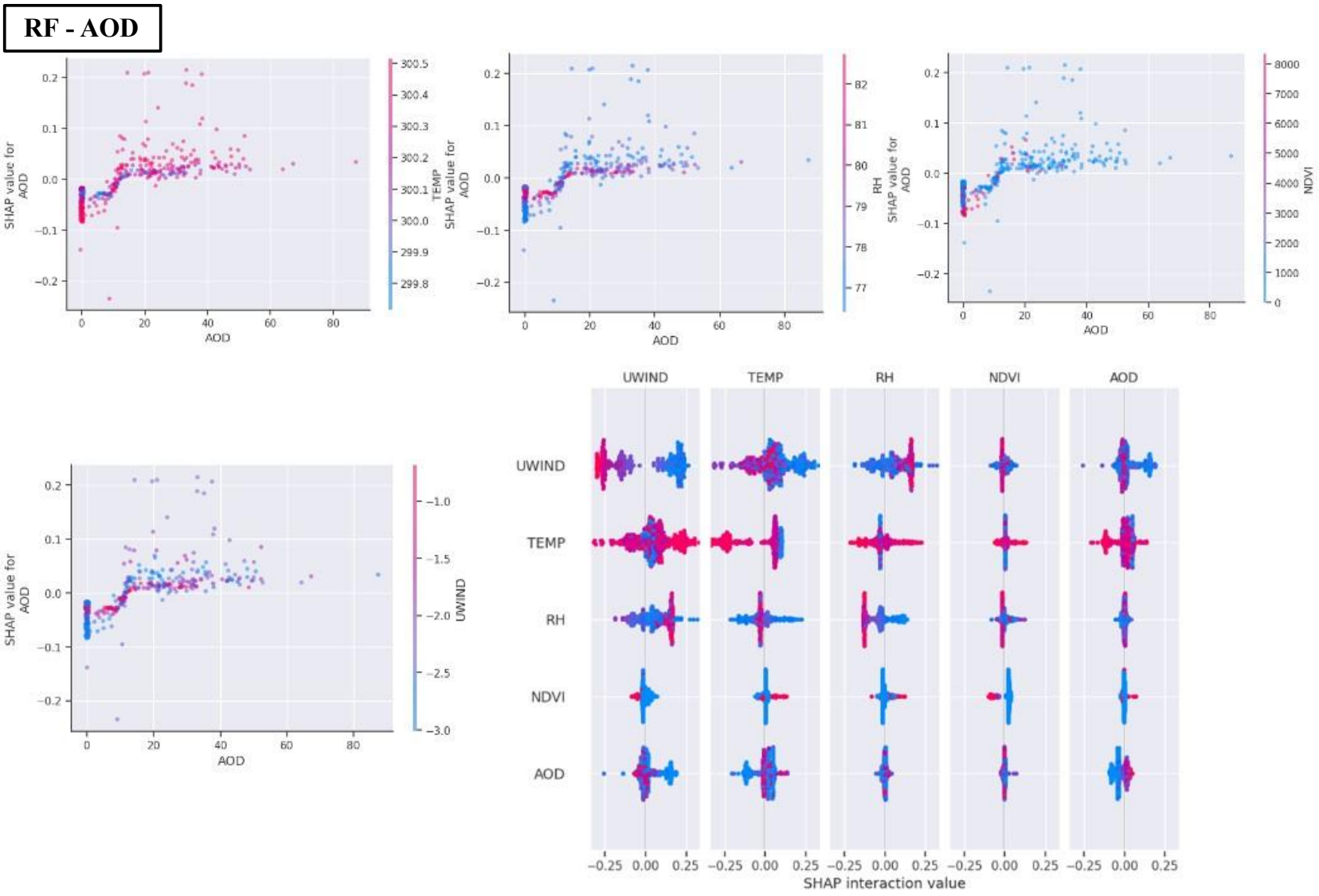

Figure S7: Showing the local interpretation comparison plot for the AOD feature in the RF method for August 2019.

GB - AOD

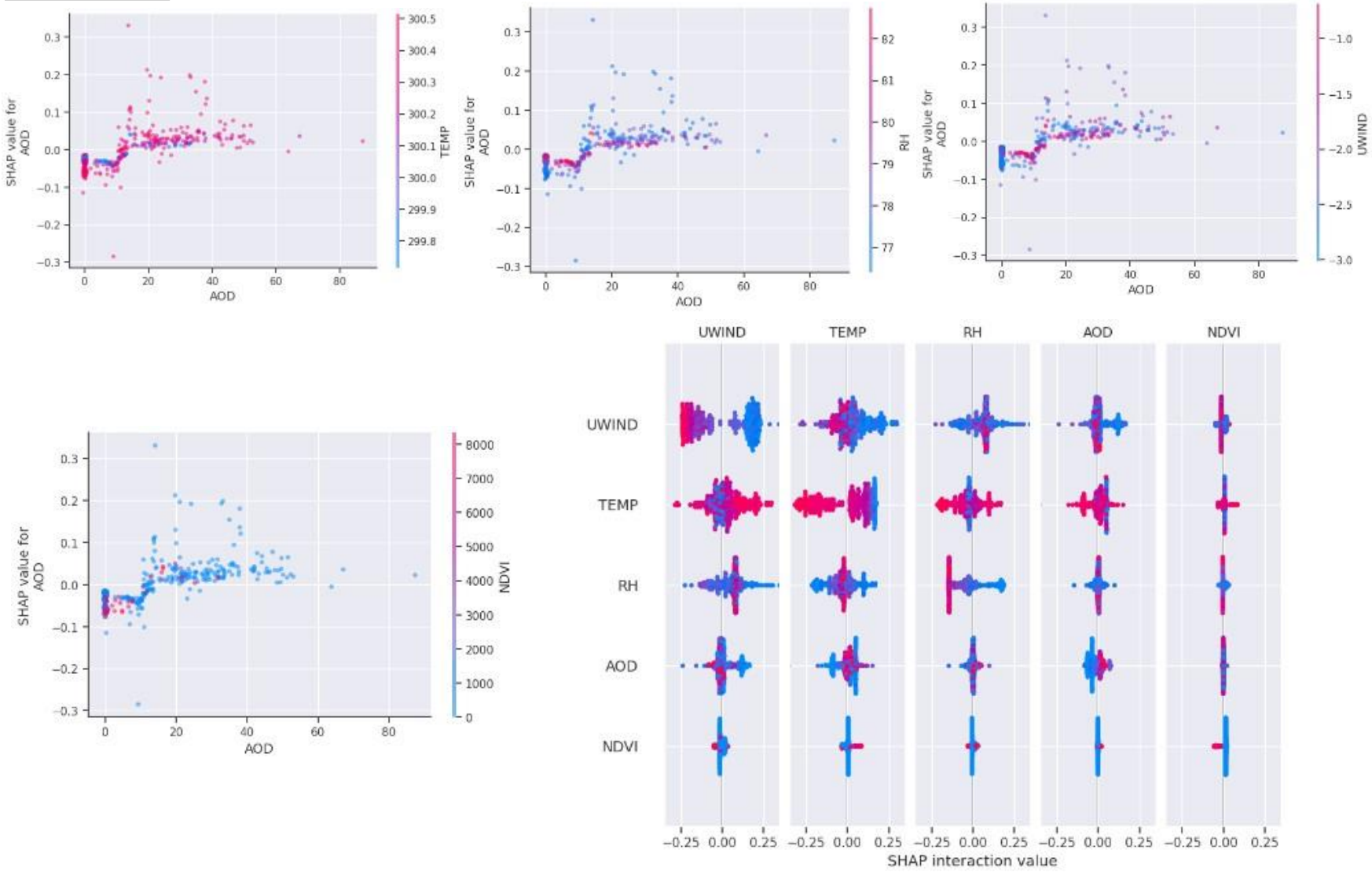

Figure S7 cont. Same as figure S7 but for the GB method.

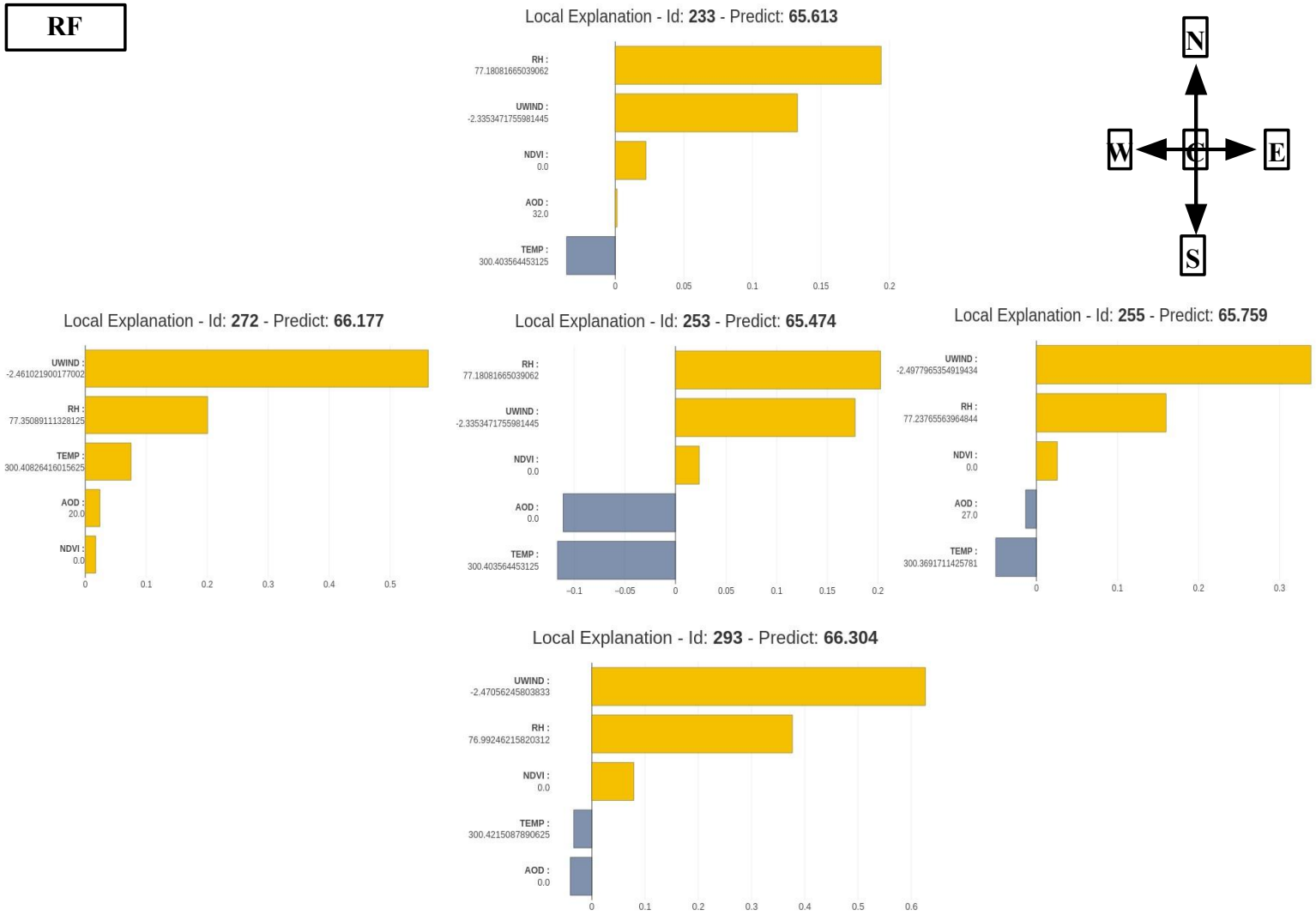

Figure S8: RF local comparison based on the direction for July 2019.

GB

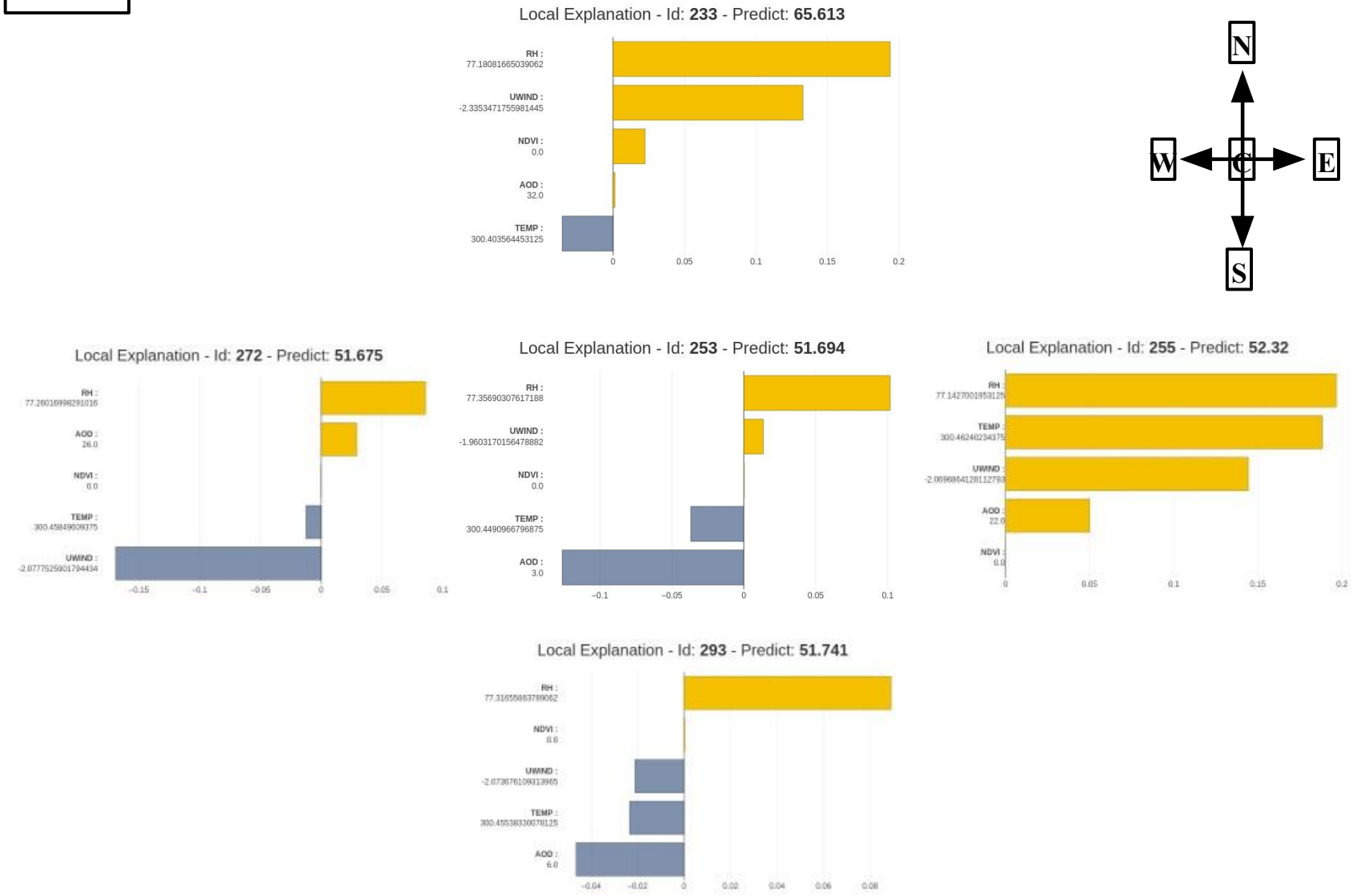

Figure S9: GB local comparison based on the direction for July 2019.

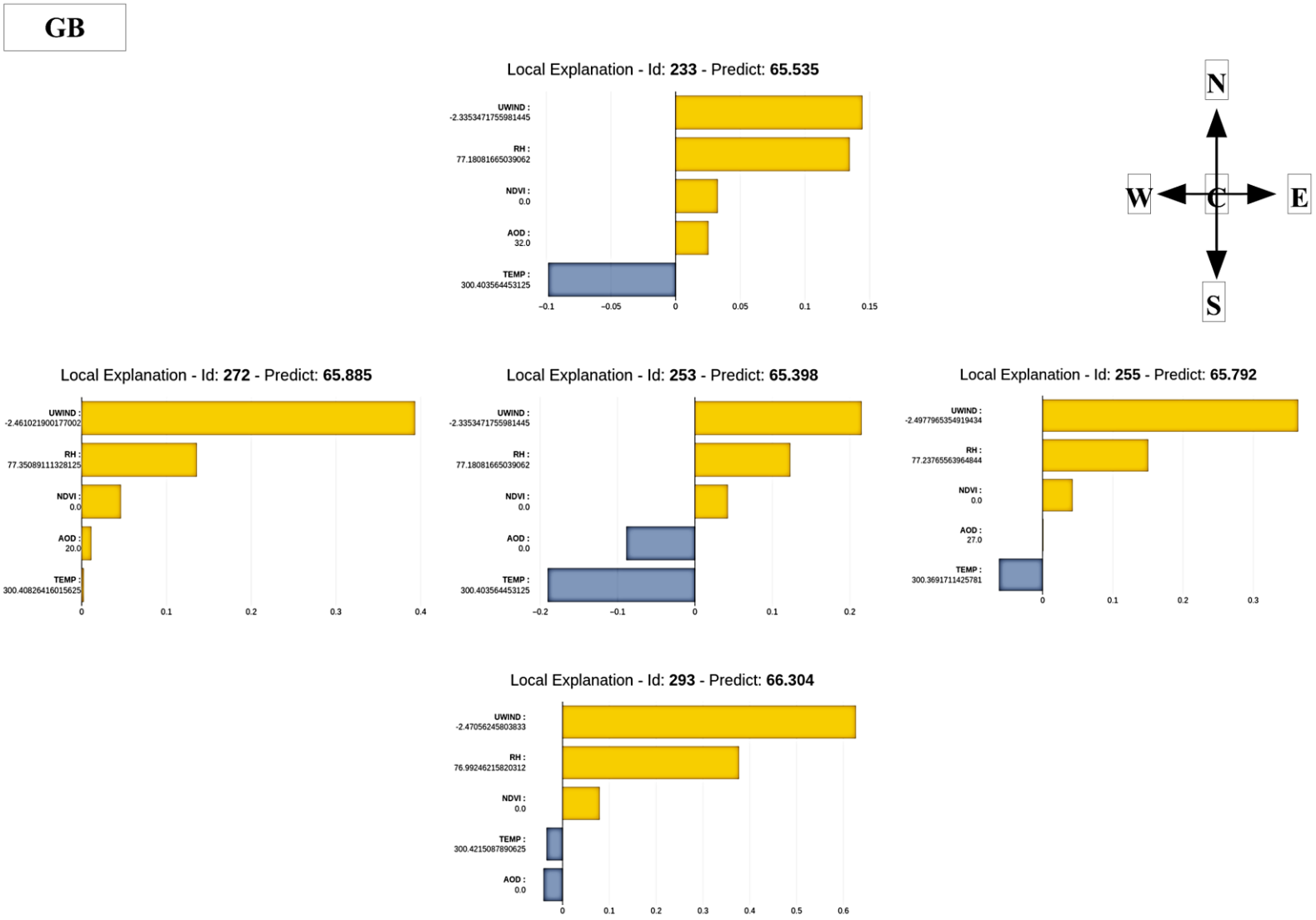

Figure S10: GB local comparison based on the direction for August 2019.

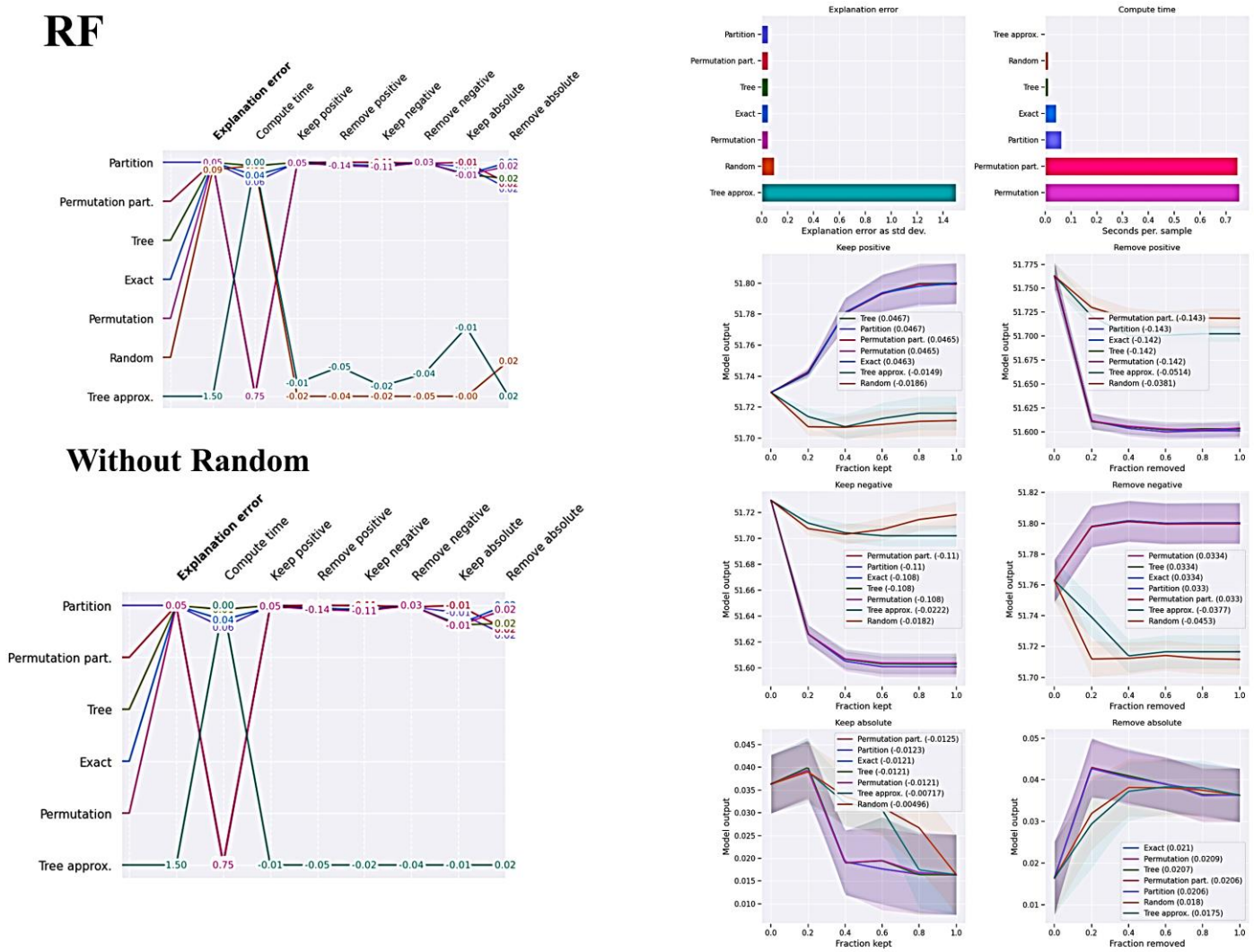

Figure S11: Displaying RF performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for July 2019.

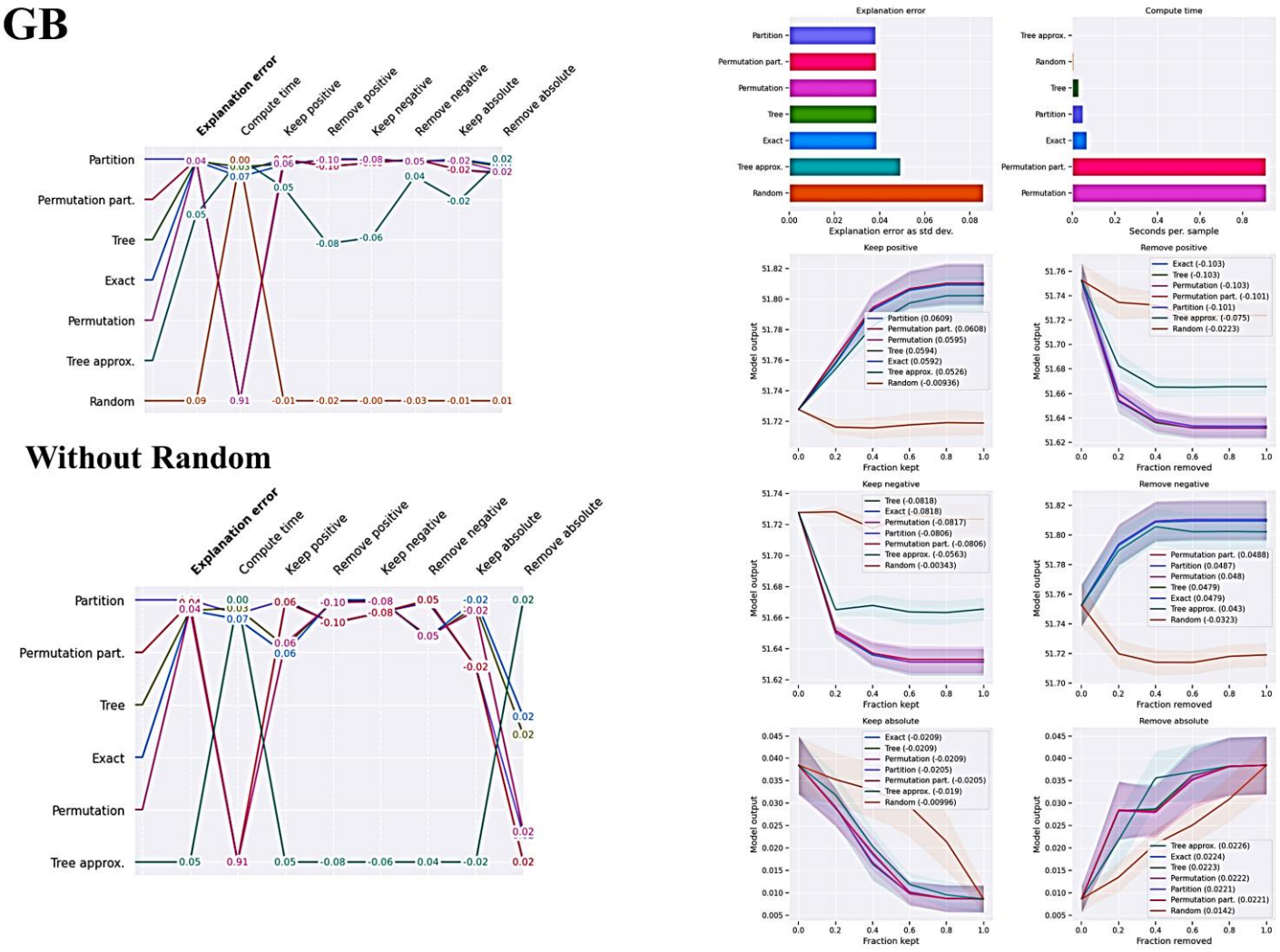

Figure S12: Displaying GB performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for July 2019.

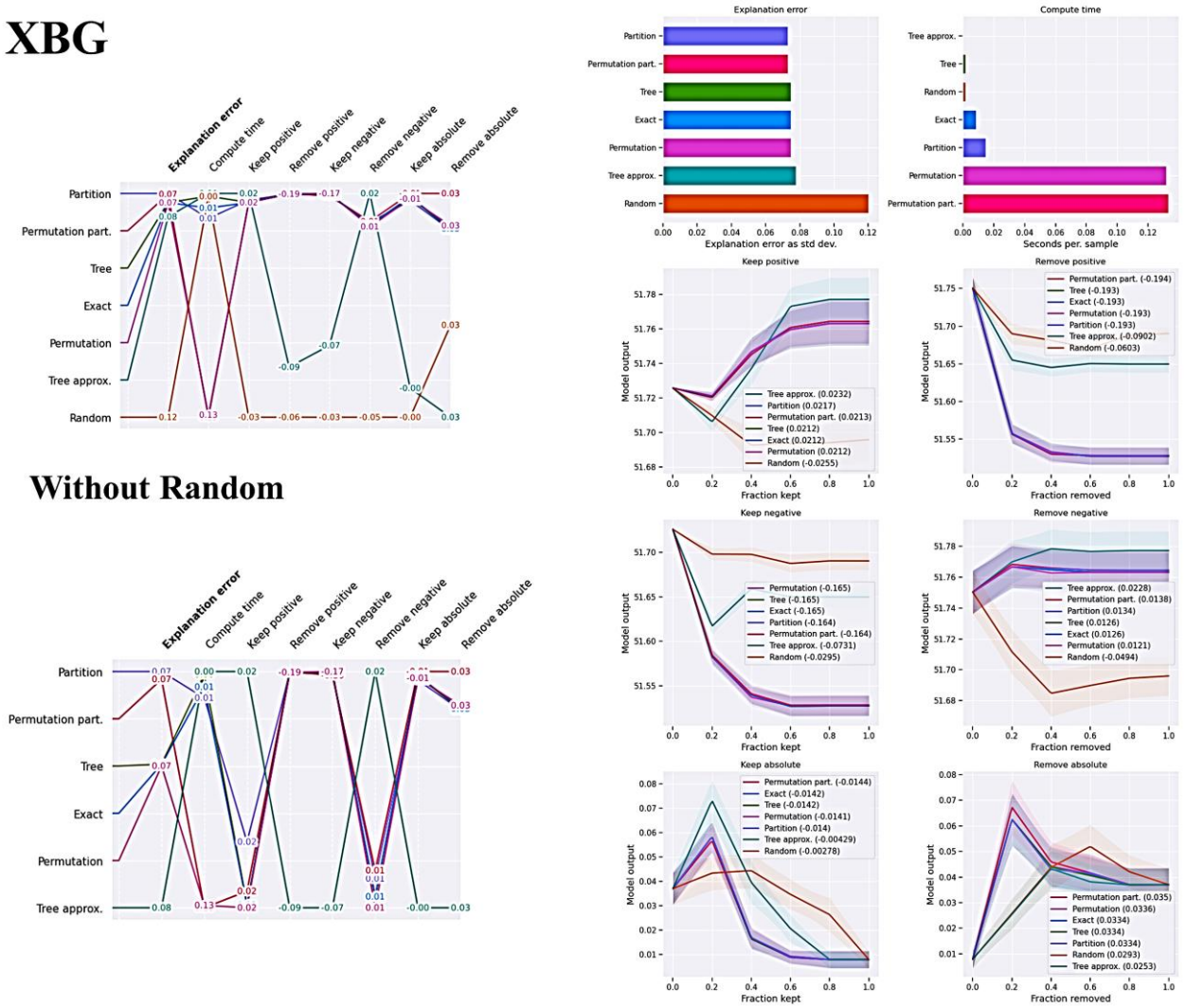

Figure S13: Displaying XGB performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for July 2019.

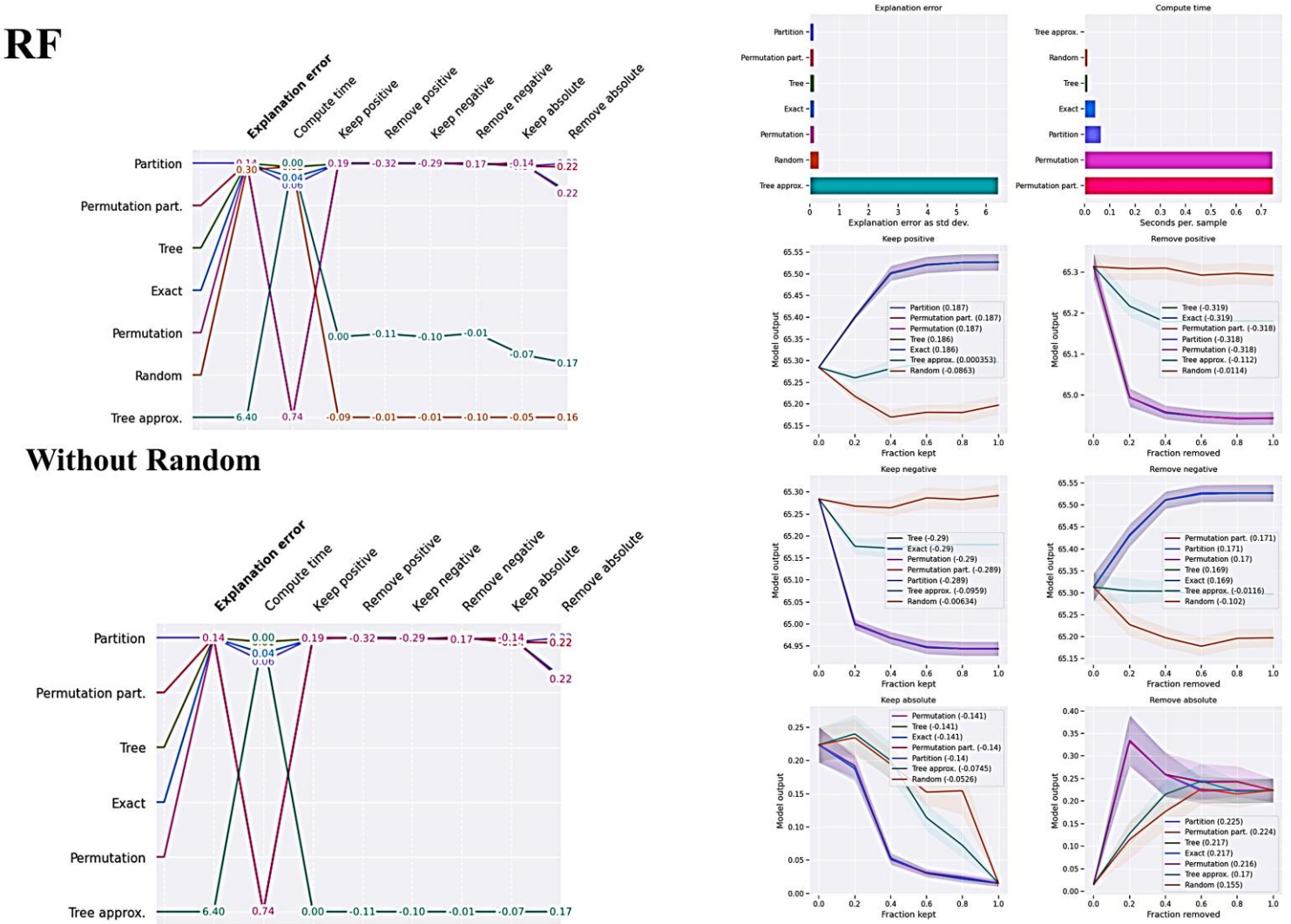

Figure S14: Displaying RF performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for August 2019.

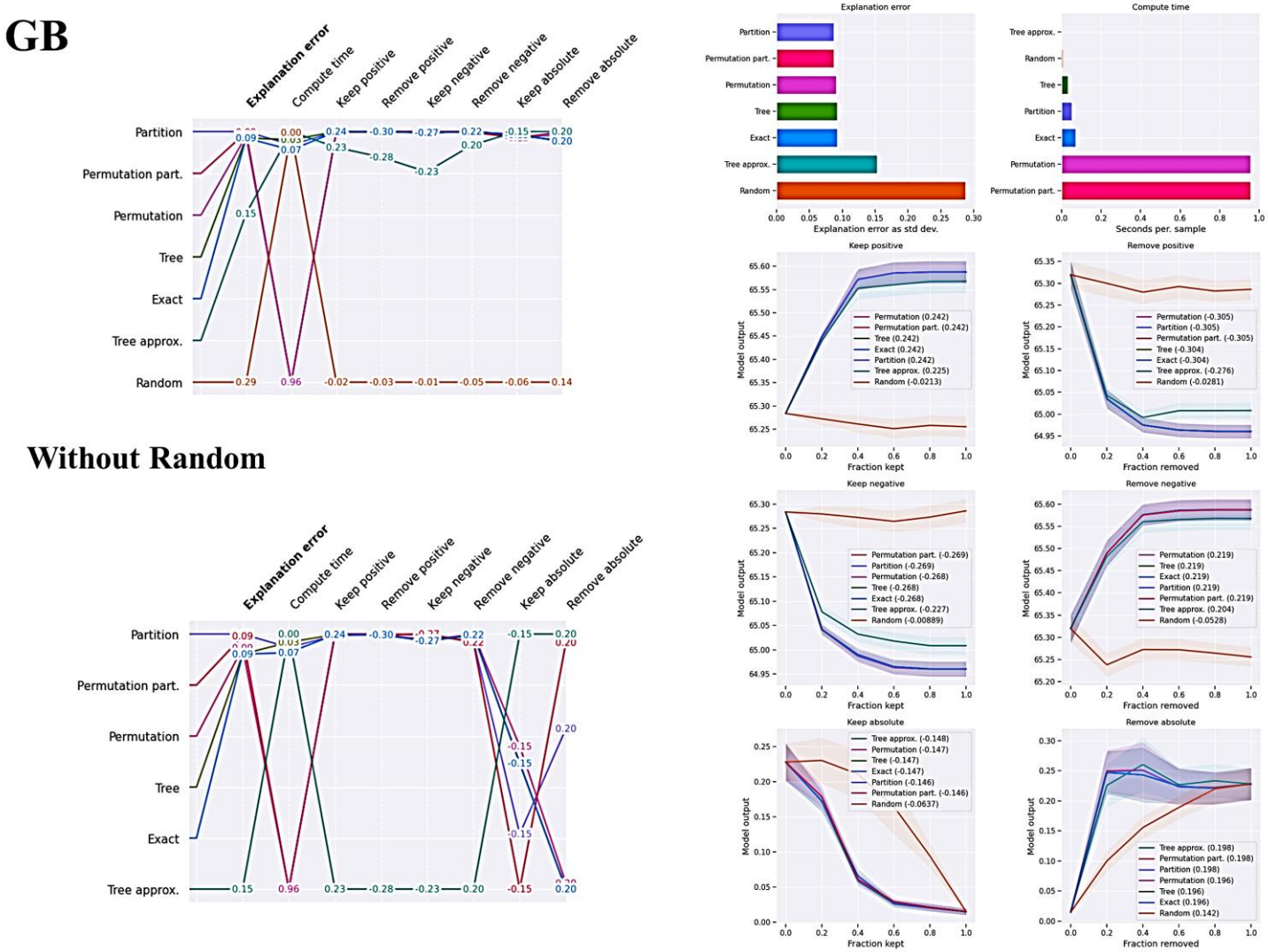

Figure S15: Displaying GB performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for August 2019.

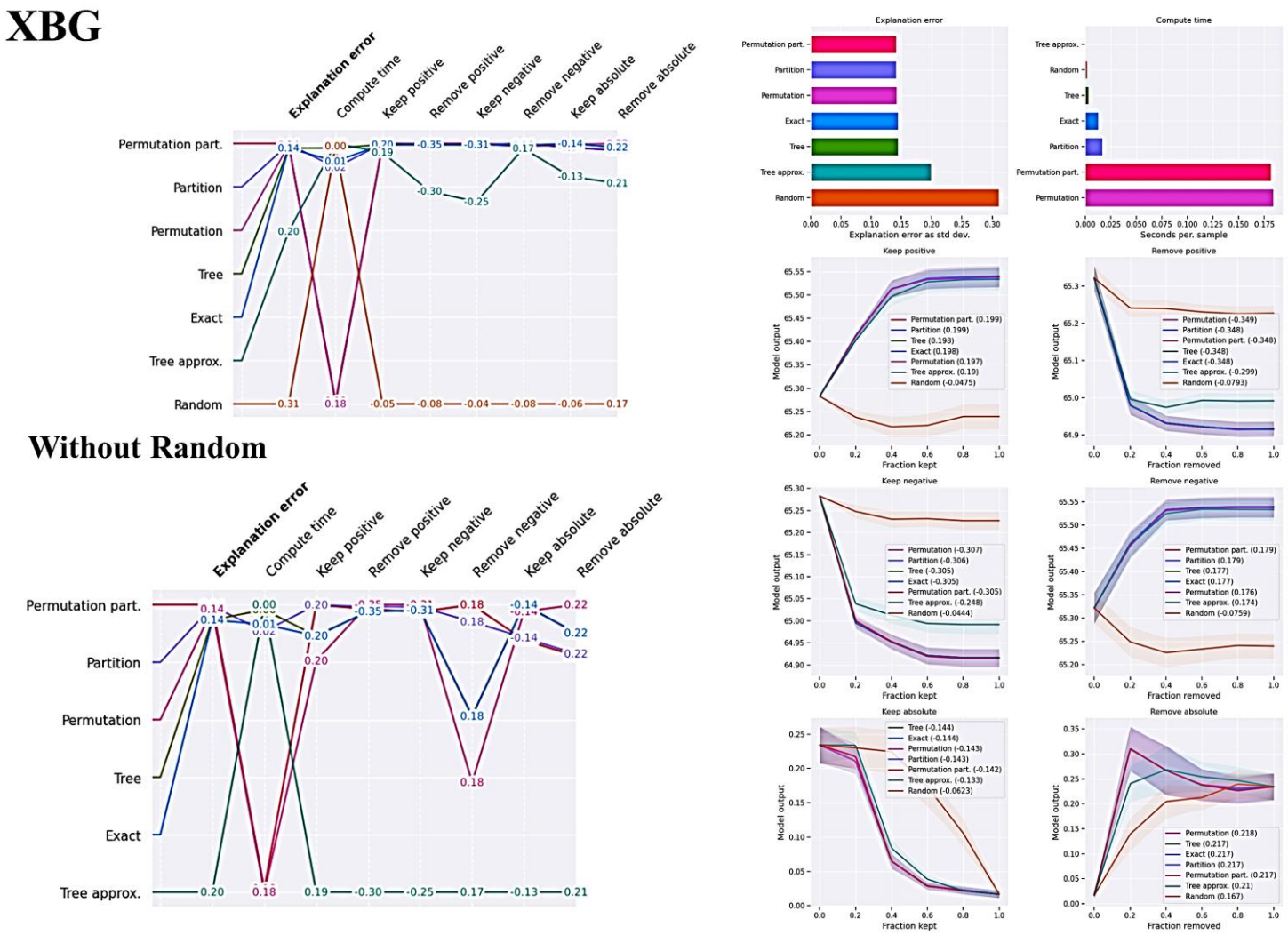

Figure S16: Displaying XGB performance graph with and without random values on the left and explanation errors, computation time, and overall model output on the right for August 2019.

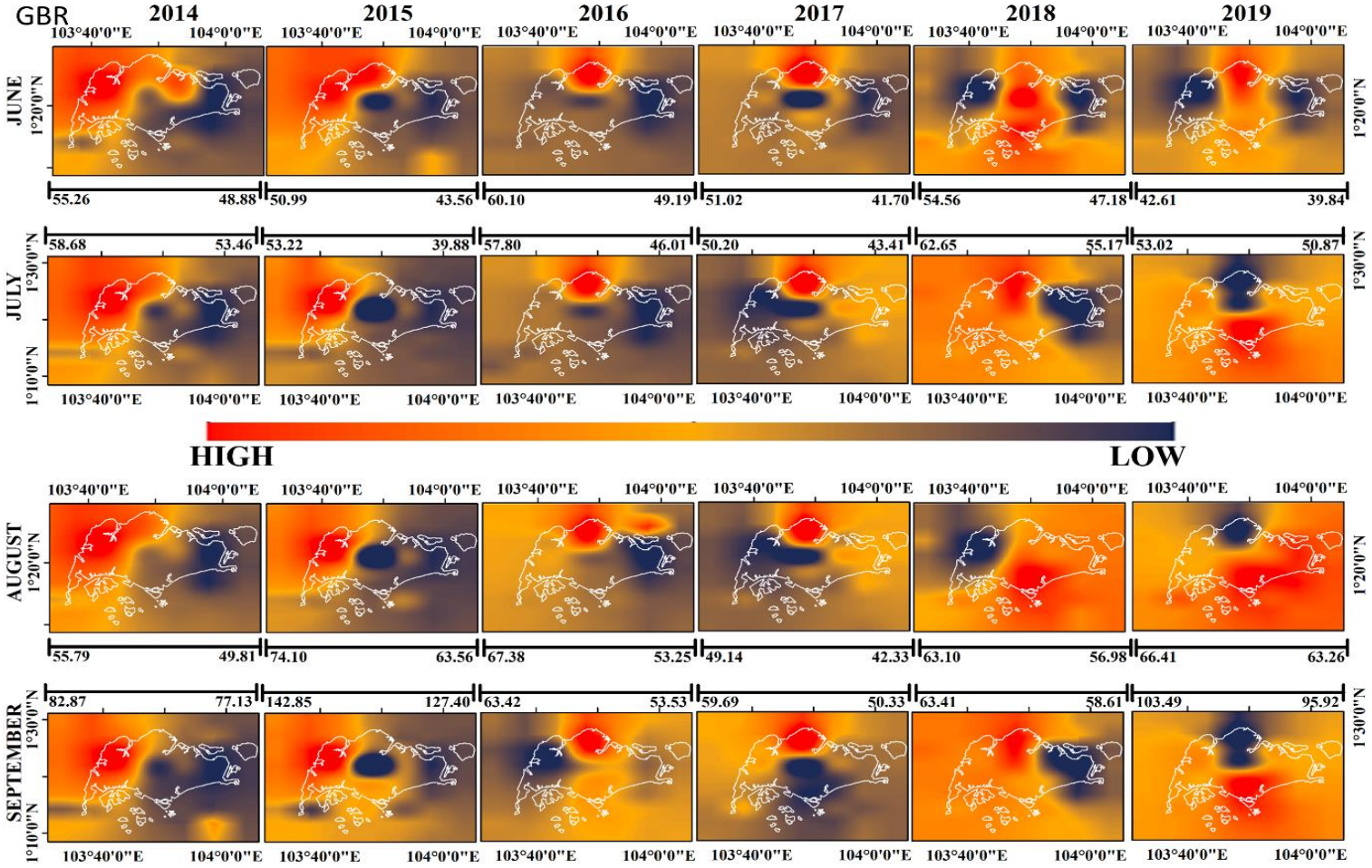

Figure S17: Spatial map using GB regression $PM_{2.5}$ concentration predictions for June–September (2014–2019).

Table S1. Best pipeline model given by TP algorithm for June – September (2014 – 2019)

Year / Months	June	July	August	September
2014	XGBRegressor(learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective="reg:squarederror", subsample=0.9500000000000001, verbosity=0)	XGBRegressor(AdaBoostRegressor(SelectFromModel(input_matrix, max_features=0.35000000000000003, n_estimators=100, threshold=0.25), learning_rate=0.01, loss=linear, n_estimators=100), learning_rate=1.0, max_depth=3, min_child_weight=3, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9000000000000001, verbosity=0)	XGBRegressor(RobustScaler(input_matrix), learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(SelectFwe(input_matrix, alpha=0.03), learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)
2015	XGBRegressor(SelectFwe(input_matrix, alpha=0.012), learning_rate=0.1, max_depth=9, min_child_weight=8, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	DecisionTreeRegressor(RandomForestRegressor(SelectFwe(input_matrix, alpha=0.03), bootstrap=False, max_features=0.5, min_samples_leaf=8, min_samples_split=17, n_estimators=100), max_depth=7, min_samples_leaf=5, min_samples_split=7)	ExtraTreesRegressor(XGBRegressor(input_matrix, learning_rate=0.1, max_depth=7, min_child_weight=6, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0), bootstrap=False, max_features=0.7500000000000001, min_samples_leaf=4, min_samples_split=15, n_estimators=100)	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=7, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)
2016	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(SelectFwe(input_matrix, alpha=0.005), learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(SelectFwe(input_matrix, alpha=0.014), learning_rate=1.0, max_depth=3, min_child_weight=3, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9000000000000001, verbosity=0)	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=10, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)
2017	XGBRegressor(DecisionTreeRegressor(RandomForestRegressor(input_matrix, bootstrap=False, x_features=0.5, min_samples_leaf=8, min_samples_split=17, n_estimators=100), max_depth=9, min_samples_leaf=11, min_samples_split=19), learning_rate=0.1, max_depth=6, min_child_weight=2, n_estimators=100, n_jobs=1,	RandomForestRegressor(input_matrix, bootstrap=False, max_features=0.7500000000000001, min_samples_leaf=4, min_samples_split=13, n_estimators=100)	RandomForestRegressor(input_matrix, bootstrap=False, max_features=0.45, min_samples_leaf=2, min_samples_split=13, n_estimators=100)	DecisionTreeRegressor(LassoLarsCV(SelectFwe(input_matrix, alpha=0.007), normalize=True), max_depth=9, min_samples_leaf=3, min_samples_split=6)

	objective=reg:squarederror, subsample=0.55, verbosity=0)			
2018	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=7, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=10, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	RandomForestRegressor(input_matrix, bootstrap=False, max_features=0.750000000000001, min_samples_leaf=2, min_samples_split=3, n_estimators=100)
2019	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=9, min_child_weight=8, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(input_matrix, learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(SelectFwe(input_matrix, alpha=0.03), learning_rate=1.0, max_depth=5, min_child_weight=8, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)	XGBRegressor(ExtraTreesRegressor(input_matrix, bootstrap=False, max_features=0.700000000000001, min_samples_leaf=2, min_samples_split=3, n_estimators=100), learning_rate=0.1, max_depth=5, min_child_weight=1, n_estimators=100, n_jobs=1, objective=reg:squarederror, subsample=0.9500000000000001, verbosity=0)