Contents

This notebook investigates Breast Cancer prediction data derived from `Ming et al.` (Gail model).

Inductive Conformal Prediction is applied to the underlying classifiers: `LogisticRegression`, `RandomForest`, `KNeighborsClassifier`, `AdaBoostClassifier`.

Nonconformity Measure `InverseProbability` is applied to the underlying classifiers (distance-based `KNNFraction` is applied separately as a reference point).

H_0 = There is no difference in the proportion of `Race` in the Lower Decile Region of predictions, based on confidence/credibility, compared to the population sample.

H_a = There is a difference in the proportion of `Race` in the Lower Decile Region of predictions, based on confidence/credibility, compared to the population sample.

In [1]:
```
import os

# set pwd to root of repository
repo_root = 'C:/Users/Bob/CHPC/conformal_prediction/vigilant-computing-machine/'

os.chdir(repo_root)
```

In [2]:
```
# 'vigilant-computing-machine/source/util.py'
import source.util as util

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns

import Orange
from Orange.distance import Euclidean
import orangecontrib.conformal as cp
```

Read in Data

In [3]:
```
tab = Orange.data.Table('./data/signal_with_header_for_orange.csv')
```

Inspect the Data

In [4]:
```
tab.domain
```

Out[4]:
```
[T1, N_Biop, HypPlas, AgeMen, Age1st, N_Rels, Race | Case_signalYN]
```

In [5]:
```
tab.domain.attributes
```

Out[5]:
```
(ContinuousVariable(name='T1', number_of_decimals=0),
 DiscreteVariable(name='N_Biop', values=('0', '1', '2', '3', '4', '5', '6')),
 DiscreteVariable(name='HypPlas', values=('0', '1', '99')),
 ContinuousVariable(name='AgeMen', number_of_decimals=0),
 ContinuousVariable(name='Age1st', number_of_decimals=0),
 DiscreteVariable(name='N_Rels', values=('0', '1', '2', '3', '4', '5', '6')),
 DiscreteVariable(name='Race', values=('1', '2', '3', '4', '5', '6', '7')))
```

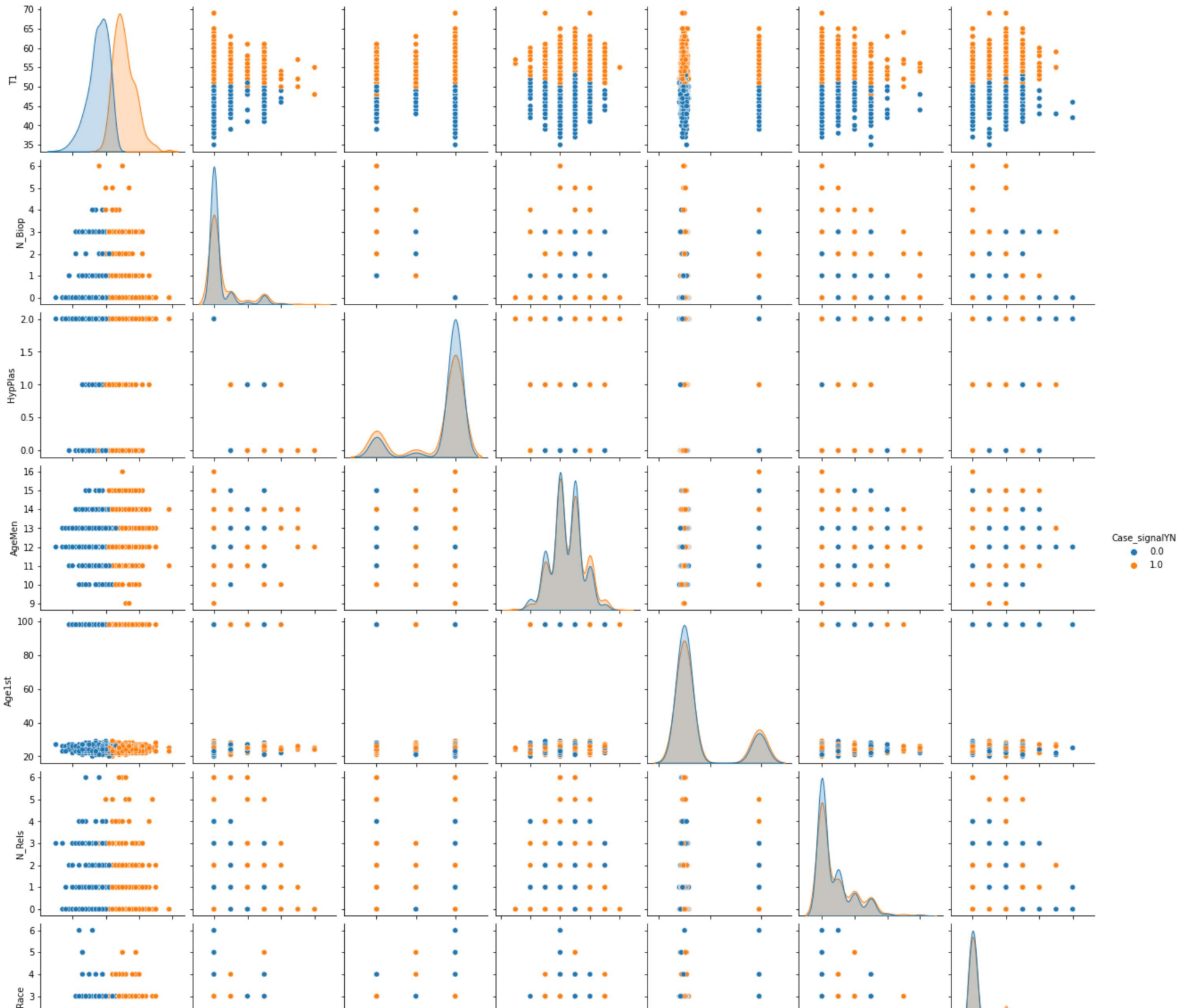
Are our data normally distributed?

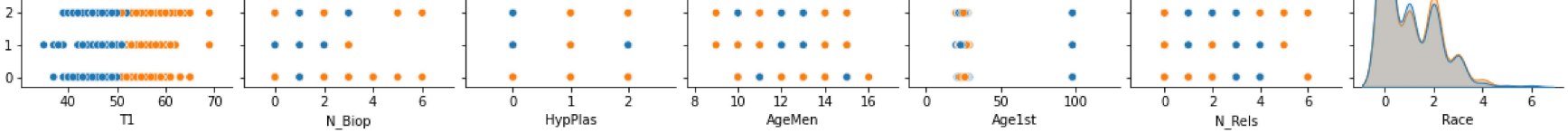
Are there any problematic values in any of our continuous variables?

In [6]:
```
df = util.table_to_df(tab, x_only=False)
sns.pairplot(df, hue='Case_signalYN')
df.describe()
```

Out[6]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
count	1200.000000	1200.00000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000	1200.000000
mean	51.575833	0.41250	1.639167	12.460000	38.780000	0.764167	0.946667	0.499167
std	5.021812	0.95818	0.741231	1.044498	29.036601	1.130463	1.114237	0.500208
min	35.000000	0.00000	0.000000	9.000000	20.000000	0.000000	0.000000	0.000000
25%	48.000000	0.00000	2.000000	12.000000	24.000000	0.000000	0.000000	0.000000
50%	52.000000	0.00000	2.000000	12.000000	25.000000	0.000000	1.000000	0.000000
75%	55.000000	0.00000	2.000000	13.000000	27.000000	1.000000	2.000000	1.000000
max	69.000000	6.00000	2.000000	16.000000	98.000000	6.000000	6.000000	1.000000


T1
N_Biop
HypPlas
AgeMen
Age1st
N_Rels
Race
Case_signalYN
0.0
1.0



Are our data normally distributed?

Our continuous variables `T1` and `AgeMen` are not horribly far from normally distribution.

Are there any problematic values in any of our continuous variables?

Our remaining continuous variable, `Age1st`, uses the value `98` to indicate nulliparious (individual has not borne offspring).

`Age1st`'s standard deviation is being exaggerated to ~29 as a result of this encoding.

This could be problematic and there are a variety of ways to go about mitigating this issue; however, for simplicity, we will take not action at this time.

Get `Race` distribution and cancer occurrence in population sample (Ming et al.)

In [7]:
```
# get distribution of race
df_race_signal = pd.DataFrame()
df_race_signal['race'] = tab.X[:,-1].astype(int)
df_race_signal_counts = df_race_signal.value_counts().sort_index()
n_signal_race = df_race_signal_counts.values.sum()

# get occurrence of cancer by race
df_race_signal['cancer'] = tab.Y.astype(int)
p_cancer_by_race = df_race_signal.groupby('race').mean('cancer')

df_race_signal_prop = pd.DataFrame({'count_n': df_race_signal_counts,
                                    'count_p': df_race_signal_counts / n_signal_race,
                                    'cancer': p_cancer_by_race.values.flatten()})
df_race_signal_prop
```

Out[7]:

race	count_n	count_p	cancer
0	589	0.490833	0.490662
1	234	0.195000	0.478632
2	260	0.216667	0.526923
3	94	0.078333	0.510638
4	17	0.014167	0.647059
5	4	0.003333	0.500000
6	2	0.001667	0.000000

Observation:

The dataset above (Ming et al.) does not appear representative of many pathologies subject to Predictive Analytics in the Clinical Sciences.

The dataset above was intended to compare modern Machine Learning techniques to the traditional Gail model.

Within the original context, the Breast Cancer prevalence of approximately 50% makes sense.

In the Clincial Sciences, a large number of pathologies subject to Predictive Analytics are much less prevalent.

Mitigation:

We will randomly remove 9/10 cancer cases; however, this may further diminish the quanitity of individuals belonging to minority races.

This will create class imbalance (10:1), which is more representative (relatively) of the prevalence of pathologies such as Breast Cancer.

Note:

We will later see that, in either case (balanced/imbalanced), the particular individuals in the less prevalent minority races happen to have risk-factor values that (when combined with the prevalence of cancer among members of the race) lead to high-confidence predictions.

Subsequently, we generate a larger population sample using the same procedure employed to create the original data (Ming et al.).

With a larger (ten-fold) sample size, we can more reasonably expect to have a viable representation of minority races, as well as being able to maintain this representation of minority races when randomly removing 9/10 cancer cases.

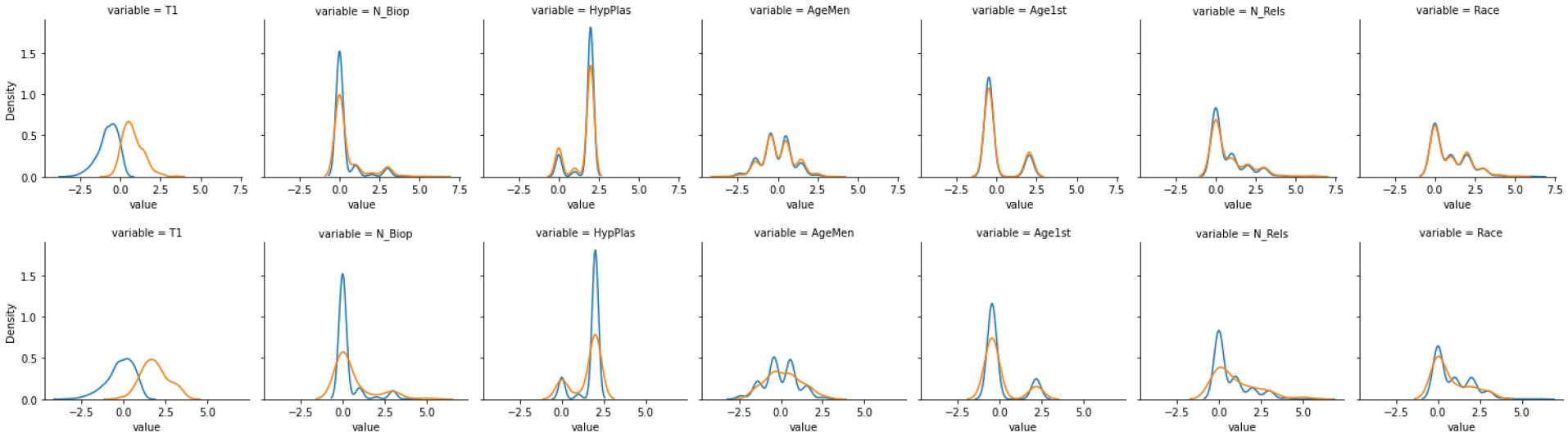
Randomly remove 9/10 cancer cases

In [8]:
```
# probability of getting breast cancer is not ~50-50

# get non-cancer
non_cancer_indices = np.array([x.row_index for x in tab if x.get_class().value == '0'])

# get cancer (randomized, but reproducible)
cancer_indices = np.array([x.row_index for x in tab if x.get_class().value == '1'])
cancer_indices_randomized = np.random.default_rng(42).permutation(cancer_indices)

# get 1 in 10 cancer
cancer_indices_tenth = cancer_indices_randomized[-int(len(cancer_indices) / 10):]

# create new table with cancer:non-cancer == 1/10:1
tab_dec = Orange.data.Table.from_table_rows(tab,
                                            np.concatenate([cancer_indices_tenth,
                                                            non_cancer_indices]))
tab.name = 'Ming et al. (1:1)'
tab_dec.name = 'Ming et al. (10:1)'
normalizer = Orange.preprocess.Normalizer(norm_type = Orange.preprocess.Normalize.NormalizeBySD)
```

Visualize 1:1 vs 10:1 distributions (normalized)

In [9]:
```
# 1:1 healthy:cancer
d = util.table_to_df(normalizer(tab), x_only=False)
d = pd.melt(d, d.columns[-1], d.columns[:-1])
```

```
g = sns.FacetGrid(d, col='variable', hue='Case_signalYN')
g.map(sns.kdeplot, 'value')
plt.show()

# 10:1 healthy:cancer
d = util.table_to_df(normalizer(tab_dec), x_only=False)
d = pd.melt(d, d.columns[-1], d.columns[:-1])
g = sns.FacetGrid(d, col='variable', hue='Case_signalYN')
g.map(sns.kdeplot, 'value')
plt.show()
```



Observation:

We can see that the 10:1 healthy:cancer data remains fairly representative of the 1:1 data.

Get `Race` distribution and cancer occurrence in population sample (10:1 healthy:cancer)

In [10]:
```
# get distribution of race
df_race_signal_dec = pd.DataFrame()
df_race_signal_dec['race'] = tab_dec.X[:,-1].astype(int)
df_race_signal_dec_counts = df_race_signal_dec.value_counts().sort_index()
n_signal_race = df_race_signal_dec_counts.values.sum()

# get occurrence of cancer by race
df_race_signal_dec['cancer'] = tab_dec.Y.astype(int)
p_cancer_by_race = df_race_signal_dec.groupby('race').mean('cancer')

df_race_signal_downsampl_prop = pd.DataFrame({'count_n': df_race_signal_dec_counts,
                                              'count_p': df_race_signal_dec_counts / n_signal_race,
                                              'cancer': p_cancer_by_race.values.flatten()})

# add race count
df_race_signal_prop['count_n_dec'] = df_race_signal_dec_counts

# add race proportion
df_race_signal_prop['count_p_dec'] = df_race_signal_dec_counts / n_signal_race

# add race cancer prevalence
```

```
df_race_signal_prop['cancer_dec'] =  p_cancer_by_race.values.flatten()

# reorder
df_race_signal_prop = df_race_signal_prop.iloc[:,[0,3,2,5,1,4]]
df_race_signal_prop
```

Out[10]:

race	count_n	count_n_dec	cancer	cancer_dec	count_p	count_p_dec
0	589	335	0.490662	0.104478	0.490833	0.507576
1	234	131	0.478632	0.068702	0.195000	0.198485
2	260	132	0.526923	0.068182	0.216667	0.200000
3	94	52	0.510638	0.115385	0.078333	0.078788
4	17	6	0.647059	0.000000	0.014167	0.009091
5	4	2	0.500000	0.000000	0.003333	0.003030
6	2	2	0.000000	0.000000	0.001667	0.003030

Note:

The small number of individuals belonging to `race` 4, 5, and 6--combined with the lack of these any of individuals having cancer (`cancer_dec`)--may be problematic and create an over-confidence bias for prections corresponding to individuals from these three races.

This is due to the population sample being unrepresentative.

Individuals belonging to `race` 4/5/6 do get cancer (`cancer_dec` , our 10:1 sample, may mislead our classifier).

This was already the case for `race` 6, prior to downsampling the cancer population within our sample.

Get Cancer Correlation with Independent Variables

In [11]:
```
df_tab = util.table_to_df(tab, x_only=False)
df_cancer_X_correlation_matrix_signal = pd.DataFrame(df_tab.corr().iloc[-1, :]).T
df_cancer_X_correlation_matrix_signal
```

Out[11]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.773642	0.121657	-0.11216	0.061395	0.046615	0.063812	0.025359	1.0

In [12]:
```
df_tab_dec = util.table_to_df(tab_dec, x_only=False)
df_cancer_X_correlation_matrix_signal_dec = pd.DataFrame(df_tab_dec.corr().iloc[-1, :]).T
df_cancer_X_correlation_matrix_signal_dec
```

Out[12]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.584841	0.106319	-0.088406	0.055188	-0.004393	0.081889	-0.040239	1.0

Observation:

Majority of the correlation coefficients retain their respective proportions and sign change only occurred on the two least correlated variables.

This is not to say that correlation coefficients alone quantify how similar/different two given samples are, but they can act as a canary in a coal mine.

As noted above, we see the lack of any individuals belonging to `race` 4/5/6 having cancer may be contributing to the correlation between `Race` and `Case_signalYN` (cancer) becoming slightly negative.

This is a fairly reasonable result and we shall proceed.

Subsequently, we repeat the following experiments with a larger dataset in order to address the unrepresentative-ness noted previously.

Experiment:

We will utilize Logistic Regression as our underlying classifier.

We will pair this classifier with a general-purpose nonconformity measure (`InverseProbability`) $1 - p$, where p is the probability assigned to the actual class by the underlying classifier.

This pairing will be used to construct an Inductive Conformal Predictor (`InductiveClassifier`).

An Inductive Conformal Predictor will be constructed for each dataset above (healthy:cancer - 1:1 and 10:1).

Each Inductive Conformal Predictor will be fit, calibrated, and used to make predictions.

We will then evaluate the relationship between the lowest `confidence` / `credibility` predictions, respectively, and `race` .

Ming et al. Data

(original n=1200; cancer-downsample n=660)

Logistic Regression (split)

In this split experiment, we will:

- Split both sets of data (healthy:cancer = 1:1 & 10:1) into train/calibrate/test sets
- Normalize each split individually
- Explore the high variance among splits' cancer correlation coefficients
- Run a condensed form of the experiment
- Assess the results/issues
- Propose a way forward

Splits data into train, calibrate, and test sets

1:1 healthy:cancer

In [13]:
```
np.random.seed(42) # reproducibility

train, test = next(cp.evaluation.RandomSampler(tab, 2, 1)) # 2:1 train:test
train, calibrate = next(cp.evaluation.RandomSampler(train,2,1)) # 2:1 train:calibrate
```

10:1 healthy:cancer

In [14]:
```
np.random.seed(42) # reproducibility

train_dec, test_dec = next(cp.evaluation.RandomSampler(tab_dec, 2, 1)) # 2:1 train:test
train_dec, calibrate_dec = next(cp.evaluation.RandomSampler(train_dec,2,1)) # 2:1 train:calibrate
```

Normalize splits

Note: one should review data after splitting to validate the type of normalization used; however, substantial bias relating to aspects of categorical variables (e.g., `race`) that are not subject to normalization is already anticipated.

In [15]:
```
normalizer = Orange.preprocess.Normalizer(norm_type = Orange.preprocess.Normalize.NormalizeBySD)

train = normalizer(train)
calibrate = normalizer(calibrate)
test = normalizer(test)

train_dec = normalizer(train_dec)
calibrate = normalizer(calibrate_dec)
test_dec = normalizer(test_dec)
```

Check cancer correlation coefficients among splits (1:1 healthy:cancer)

In [16]:
```
# train 1:1
pd.DataFrame(util.table_to_df(train, x_only=False).corr().Case_signalYN).T
```

Out[16]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.782313	0.094094	-0.098833	0.061348	0.040824	0.029199	0.042755	1.0

In [17]:
```
# calibrate 1:1
pd.DataFrame(util.table_to_df(calibrate, x_only=False).corr().Case_signalYN).T
```

Out[17]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.552471	0.145558	-0.170856	0.117198	0.041634	0.078212	-0.194677	1.0

In [18]:
```
# test 1:1
pd.DataFrame(util.table_to_df(test, x_only=False).corr().Case_signalYN).T
```

Out[18]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.768433	0.169719	-0.136046	0.070738	0.033485	0.095501	-0.011524	1.0

Split Variance Issue

Observation:

We can see the impact that splitting the data has had on the cancer correlation coefficients is quite drastic, particularly when compared to the impact we observed in randomly removing 9/10 cancer cases.

We can clearly see that variance among the train/calibrate/test sets is quite high.

Check cancer correlation coefficients among splits (10:1 healthy:cancer)

In [19]:
```
# train 10:1
pd.DataFrame(util.table_to_df(train_dec, x_only=False).corr().Case_signalYN).T
```

Out[19]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.595656	0.003041	-0.013202	0.020844	-0.032689	0.072159	0.037647	1.0

In [20]:
```
# calibrate 10:1
pd.DataFrame(util.table_to_df(calibrate_dec, x_only=False).corr().Case_signalYN).T
```

Out[20]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.552471	0.145558	-0.170856	0.117198	0.041634	0.078212	-0.194677	1.0

In [21]:
```
# test 10:1
pd.DataFrame(util.table_to_df(test_dec, x_only=False).corr().Case_signalYN).T
```

Out[21]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.594393	0.217395	-0.141402	0.046391	0.005259	0.098323	-0.032214	1.0

Observation:

We again see similar results and it is quite clear that variance among the train/calibrate/test sets is quite high.

The exception being the correlation between `T1` and `Case_signalYN` (cancer), not surprisingly, as we have removed 9/10 cancer cases.

Run condensed experiment

In [22]:
```
np.random.seed(42)

# create learner
cc = cp.classification.InductiveClassifier(
        cp.nonconformity.InverseProbability(
```

```
            Orange.classification.LogisticRegressionLearner()))

# run experiments
experiment_logistic_signal = util.run_experiments(cc, train, test, calibrate, eps=0.05, method='Baseline Split Data')
experiment_logistic_cancer_downsampl = util.run_experiments(cc, train_dec, test_dec, calibrate_dec, eps=0.05, method='Baseline Split Data')

# get LDR (low confidence predictions) for 1:1
df_pred = experiment_logistic_signal.loc[0, 'df']
df_pred = util.sort_reindex(df_pred, col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_signal_ldr = df_pred.iloc[-ldr_idx:,:].copy()

# get LDR (low confidence predictions) for 10:1
df_pred = experiment_logistic_cancer_downsampl.loc[0, 'df']
df_pred = util.sort_reindex(df_pred, col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_signal_cancer_downsampl_ldr = df_pred.iloc[-ldr_idx:,:].copy()
```

`Race` distribution in LDR

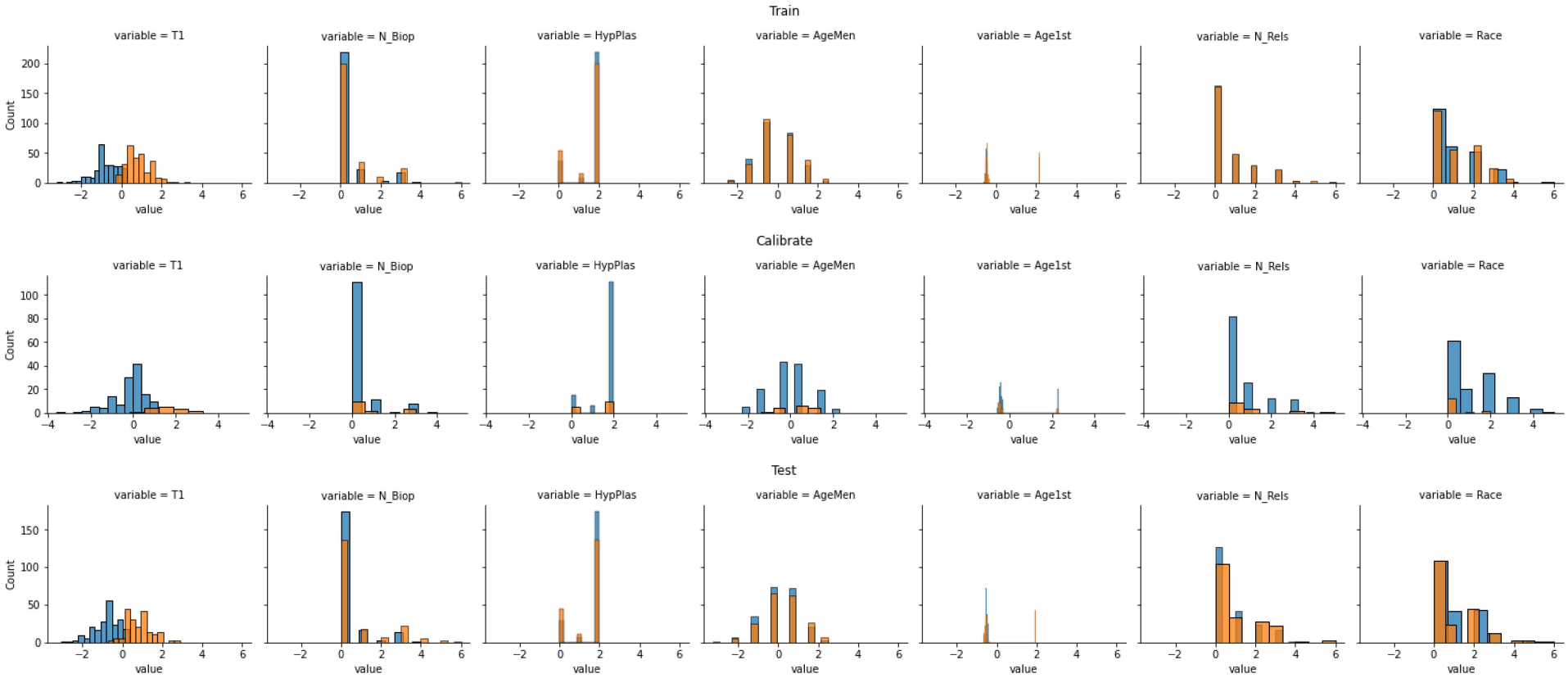
Visualize representative-ness of 1:1 healthy:cancer dataset splits by class

---

Build table of 1:1 and 10:1 low confidence predictions and `Race` distribution

In [23]:
```
# Experiment
# 1:1 healthy:cancer (split)
counts = df_pred_logistic_signal_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_split': counts / sums,
                            'ldr_n_split': counts})
table = df_race_signal_prop.join(proportions,
                                 on='race',
                                 how='left')
table = table.fillna(0)
table['ldr_n_split'] = table['ldr_n_split'].astype(int)
df_logistic_ldr_race_disparity = table.copy()
df_logistic_ldr_race_disparity = df_logistic_ldr_race_disparity.iloc[:, [0,1,7,2,3,4,5,6]]


# visualize how representative the 1:1 healthy:cancer data's train/test/calibration splits are
# train - 1:1
d = util.table_to_df(train, x_only=False)
d = pd.melt(d, d.columns[-1], d.columns[:-1])
g = sns.FacetGrid(d, col='variable', hue='Case_signalYN')
g.map(sns.histplot, 'value')
g.fig.subplots_adjust(top=0.8)
g.fig.suptitle('Train')
plt.show()

# calibrate - 1:1
d = util.table_to_df(calibrate, x_only=False)
d = pd.melt(d, d.columns[-1], d.columns[:-1])
g = sns.FacetGrid(d, col='variable', hue='Case_signalYN')
g.map(sns.histplot, 'value')
g.fig.subplots_adjust(top=0.8)
g.fig.suptitle('Calibrate')
plt.show()
```

```
# test - 1:1
d = util.table_to_df(test, x_only=False)
d = pd.melt(d, d.columns[-1], d.columns[:-1])
g = sns.FacetGrid(d, col='variable', hue='Case_signalYN')
g.map(sns.histplot, 'value')
g.fig.subplots_adjust(top=0.8)
g.fig.suptitle('Test')
plt.show()

# Experiment
# 10:1 healthy:cancer (split)
counts = df_pred_logistic_signal_cancer_downsampl_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_dec_split': counts / sums,
                            'ldr_n_dec_split': counts})
table = df_logistic_ldr_race_disparity.join(proportions,
                                            on='race',
                                            how='left')
table = table.fillna(0)
table['ldr_n_dec_split'] = table['ldr_n_dec_split'].astype(int)
df_logistic_table = table.copy()
df_logistic_table = df_logistic_table.iloc[:, [0,1,2,9,3,4,5,6,7,8]]
df_logistic_table
```



Out[23]:

race	count_n	count_n_dec	ldr_n_split	ldr_n_dec_split	cancer	cancer_dec	count_p	count_p_dec	ldr_p_split	ldr_p_dec_split
0	589	335	27	6	0.490662	0.104478	0.490833	0.507576	0.675	0.272727
1	234	131	0	8	0.478632	0.068702	0.195000	0.198485	0.000	0.363636
2	260	132	12	2	0.526923	0.068182	0.216667	0.200000	0.300	0.090909
3	94	52	0	6	0.510638	0.115385	0.078333	0.078788	0.000	0.272727
4	17	6	0	0	0.647059	0.000000	0.014167	0.009091	0.000	0.000000
5	4	2	1	0	0.500000	0.000000	0.003333	0.003030	0.025	0.000000
6	2	2	0	0	0.000000	0.000000	0.001667	0.003030	0.000	0.000000

False Confidence

Observation:

From the rows of graphs (train/calibrate/test, respectively) of the 1:1 (healthy:cancer) sample, we can clearly see partitions vary quite drastically--particularly in their representation of the class `cancer` (orange).

Based on the table of both samples/predictions, we see no apparent pattern between the 1:1 vs 10:1 experiments. The representation of `race` in the LDR tends to be either drastically low/high or not present at all.

In both experiments, the sparser minority races appear to be more likely to **not be represented at all**. The situation is worse than **low confidence**, it is **false confidence.**

A **small number of individuals** belonging to a `race`, particularly in a situation where these indviduals do **not represent that population**, may be expected to result in either **low confidence** or **false confidence.**

Partitioning the data not only **introduces bias**, but also tends to make unrepresentative data worse.

Additionally, partitioning data **makes it possibile** for an **entire group** (e.g., `race`) to be **excluded** from the **test set.**

For example, if no individuals belonging to a given `race` are included in the test set, no predictions will be made for this `race`.

Consequently, there are no low confidence predictions and the `race` is **not be represented at all** in the LDR, predisposing one to the unfounded impression that confidence is not lacking in the predictions for a `race`.

Two options:

- Perform *repeated cross-validation*, such as *k-fold cross-validation*
- Use all individuals in the dataset for training, calibration, and testing

Because we are interested in the relationship between individual Conformal Predictions and the variables (particularly those not following a uniform distribution) upon which those predictions are made, we want to choose the option that provides the clearest view of this relationship.

A random partition of data is not necessarily balanced for all covariates. We could use stratification and attempt to create representative splits, but even with these techniques we are likely to miss latent feature space, etc.

We are not primarily interested in the suitability of a given learner for a particular problem domain or the mean accuracy of a predictive model on a set of data; however, if this were the case, *repeated cross-validation* and *leave-one-out cross-validation*, respectively, may be more suitable candidates. [1]

1. Vanwinckelen, Gitte (2 October 2019). On Estimating Model Accuracy with Repeated Cross-Validation. lirias.kuleuven. pp. 39–44. ISBN 9789461970442.

Mitigation:

Due to the **bias** incurred by splitting the data, the **high variance** of results from different splits, and the possibility of **under-representation/exclusion** of populations of interest, we will proceed by using all individuals in the dataset for training, calibration, and testing.

In doing so, we are able to more clearly focus on Conformal Prediction attributes (primarily `confidence`) and how they vary with respect to features (primarily `race`) of non-uniform distribution.

Normalize datasets that will be used as a whole

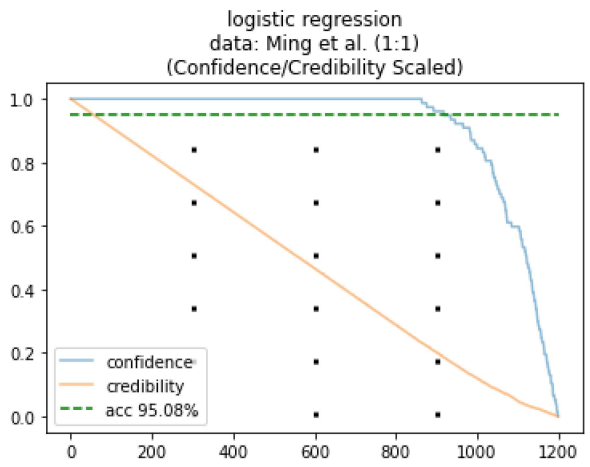
In [24]:
```
tab_signal = normalizer(tab)
tab_downsampl = normalizer(tab_dec)
```

Logistic Regression (unsplit)

In [25]:
```
np.random.seed(42)


cc = cp.classification.InductiveClassifier(
        cp.nonconformity.InverseProbability(
            Orange.classification.LogisticRegressionLearner()))

experiment_logistic_signal = util.run_experiments(cc,
                                                  tab_signal,
                                                  tab_signal,
                                                  tab_signal,
                                                  eps=0.05,
                                                  method='Baseline Unsplit Data')

util.plot_experiments(experiment_logistic_signal)
experiment_logistic_signal.iloc[:,:-3]
```

Out[25]:

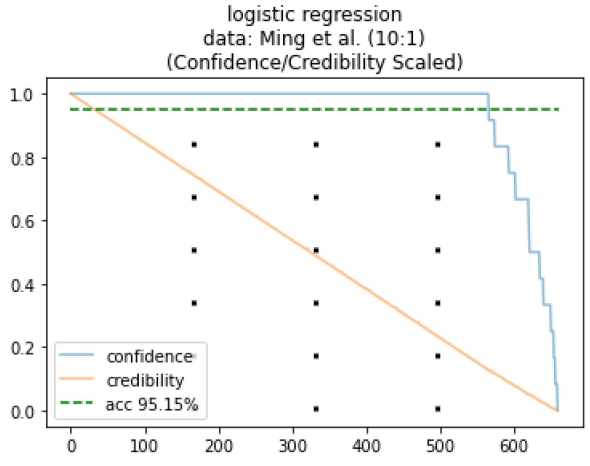
	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier	model	conformal_predictor	nonconformity
0	0.993566	0.506468	0.950833	0.0	0.97	0.920833	0.03	Ming et al. (1:1)	False	logistic regression	LogisticRegressionClassifier(skl_model=Logisti...	InductiveClassifier (InverseProbability (logis...	InverseProbability (logistic regression)



In [26]:
```
np.random.seed(42)


cc = cp.classification.InductiveClassifier(
        cp.nonconformity.InverseProbability(
            Orange.classification.LogisticRegressionLearner()))

experiment_logistic_cancer_downsampl = util.run_experiments(cc,
                                                            tab_downsampl,
                                                            tab_downsampl,
                                                            tab_downsampl,
                                                            eps=0.05,
                                                            method='Baseline Unsplit Data')

util.plot_experiments(experiment_logistic_cancer_downsampl)
experiment_logistic_cancer_downsampl.iloc[:,:-3]
```

Out[26]:

	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier	model	conformal_predictor	nonconformity
0	0.997456	0.502686	0.951515	0.039394	0.960606	0.951515	0.0	Ming et al. (10:1)	False	logistic regression	LogisticRegressionClassifier(skl_model=Logisti...	InductiveClassifier (InverseProbability (logis...	InverseProbability (logistic regression)



Observation:

In the cancer-downsample experiment, `confidence` continued to be sustained longer and fall more sharply, as expected, than the (1:1 healthy:cancer) Ming et al. dataset experiment.

Lower Decile Range (LDR) Predictions - `confidence` / `credibility`

In [27]:
```
df_pred = experiment_logistic_signal.loc[0, 'df']
df_pred = util.sort_reindex(df_pred,
                                        col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_signal_ldr = df_pred.iloc[-ldr_idx:,:].copy()
df_pred_logistic_signal_ldr.tail(1)
```

Out[27]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
1199	[0, 1]	0.935054	0.064946	0.05	[(0.0649458784346378, 0), (0.0649458784346378,...	True	0.0845	0	2	0.51721	-0.336957	0	1	1

In [28]:
```
df_pred = experiment_logistic_cancer_downsampl.loc[0, 'df']
df_pred = util.sort_reindex(df_pred,
                                                col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_signal_cancer_downsampl_ldr = df_pred.iloc[-ldr_idx:,:].copy()
df_pred_logistic_signal_cancer_downsampl_ldr.tail(1)
```

Out[28]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
659	[]	0.980333	0.019667	0.05	[(0.019667170953101363, 0), (0.019667170953101...	False	1.445904	0	2	-1.38651	-0.55208	1	0	1

`Race` distribution in LDR

In [29]:
```
# 1:1 healthy:cancer (unsplit)
counts = df_pred_logistic_signal_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_unsplit': counts / sums,
                            'ldr_n_unsplit': counts})
table = df_logistic_table.join(proportions,
                               on='race',
                               how='left')
table = table.fillna(0)
table['ldr_n_unsplit'] = table['ldr_n_unsplit'].astype(int)
df_logistic_ldr_race_disparity_unsplit = table.copy()
df_logistic_table = df_logistic_ldr_race_disparity_unsplit.iloc[:, [0,1,2,3,11,4,5,6,7,8,9,10]]

# 10:1 healthy:cancer (unsplit)
counts = df_pred_logistic_signal_cancer_downsampl_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_dec_unsplit': counts / sums,
                            'ldr_n_dec_unsplit': counts})
table = df_logistic_table.join(proportions,
                               on='race',
                               how='left')
table = table.fillna(0)
table['ldr_n_dec_unsplit'] = table['ldr_n_dec_unsplit'].astype(int)
df_logistic_signal_cancer_downsampl_ldr_race_disparity_unsplit = table.copy()
```

```
df_logistic_table = df_logistic_signal_cancer_downsampl_ldr_race_disparity_unsplit.iloc[:, [0,1,2,3,4,13,5,6,7,8,9,10,11,12]]
df_logistic_table
```

Out[29]:

race	count_n	count_n_dec	ldr_n_split	ldr_n_dec_split	ldr_n_unsplit	ldr_n_dec_unsplit	cancer	cancer_dec	count_p	count_p_dec	ldr_p_split	ldr_p_dec_split	ldr_p_unsplit	ldr_p_dec_unsplit
0	589	335	27	6	50	39	0.490662	0.104478	0.490833	0.507576	0.675	0.272727	0.416667	0.590909
1	234	131	0	8	33	13	0.478632	0.068702	0.195000	0.198485	0.000	0.363636	0.275000	0.196970
2	260	132	12	2	25	5	0.526923	0.068182	0.216667	0.200000	0.300	0.090909	0.208333	0.075758
3	94	52	0	6	12	8	0.510638	0.115385	0.078333	0.078788	0.000	0.272727	0.100000	0.121212
4	17	6	0	0	0	1	0.647059	0.000000	0.014167	0.009091	0.000	0.000000	0.000000	0.015152
5	4	2	1	0	0	0	0.500000	0.000000	0.003333	0.003030	0.025	0.000000	0.000000	0.000000
6	2	2	0	0	0	0	0.000000	0.000000	0.001667	0.003030	0.000	0.000000	0.000000	0.000000

In [30]:
```
df_logistic_table.loc[:, ['count_p', 'ldr_p_split', 'ldr_p_unsplit', 'count_p_dec', 'ldr_p_dec_split', 'ldr_p_dec_unsplit']]
```

Out[30]:

race	count_p	ldr_p_split	ldr_p_unsplit	count_p_dec	ldr_p_dec_split	ldr_p_dec_unsplit
0	0.490833	0.675	0.416667	0.507576	0.272727	0.590909
1	0.195000	0.000	0.275000	0.198485	0.363636	0.196970
2	0.216667	0.300	0.208333	0.200000	0.090909	0.075758
3	0.078333	0.000	0.100000	0.078788	0.272727	0.121212
4	0.014167	0.000	0.000000	0.009091	0.000000	0.015152
5	0.003333	0.025	0.000000	0.003030	0.000000	0.000000
6	0.001667	0.000	0.000000	0.003030	0.000000	0.000000

Observation:

We can see that we get results that appear less erratic than the partitioned data experiments, but still contain false confidence for the sparsest 2-3 `race` groups.

The small quantity of individuals belonging to these races, in either case (balanced/imbalanced), in combination with the particular individuals' risk factor values and the prevalence of cancer in each race, appears to be leading to predictions that are not in the low confidence region.

The data not being representative for these minorities is leading to bias that is unlikely to generalize well in predictions for individuals of the respective populations.

Mitigation:

Subsequently, we generate a larger population sample using the same (Ming et al.) procedure.

Given a larger (ten-fold) sample, we can more reasonably expect to capture a viable representation of minority races, as well as being able to maintain this representation of minority races when randomly removing 9/10 cancer cases from the sample.

Read in Data (10x sample size)

In [31]:
```
tab_10x = normalizer(Orange.data.Table('./data/signal_10x_with_header_for_orange.csv'))
```

Get `Race` distribution and cancer occurrence in population sample

In [32]:
```
# get distribution of race
df_race_signal_10x = pd.DataFrame()
df_race_signal_10x['race'] = tab_10x.X[:,-1].astype(int)
df_race_signal_10x_counts = df_race_signal_10x.value_counts().sort_index()
n_signal_race = df_race_signal_10x_counts.values.sum()

# get occurrence of cancer by race
df_race_signal_10x['cancer'] = tab_10x.Y.astype(int)
p_cancer_by_race = df_race_signal_10x.groupby('race').mean('cancer')

df_race_signal_10x_prop = pd.DataFrame({'count_n_10x': df_race_signal_10x_counts,
                                        'count_p_10x': df_race_signal_10x_counts / n_signal_race,
                                        'cancer_10x': p_cancer_by_race.values.flatten()})
df_race_signal_10x_prop
```

Out[32]:

race	count_n_10x	count_p_10x	cancer_10x
0	6043	0.503583	0.505378
1	2397	0.199750	0.487693
2	2366	0.197167	0.502959
3	935	0.077917	0.499465
4	126	0.010500	0.436508
5	68	0.005667	0.352941
6	65	0.005417	0.476923

Randomly remove 9/10 cancer cases

In [33]:
```
# probability of getting breast cancer is not ~50-50

# get non-cancer
non_cancer_indices = np.array([x.row_index for x in tab_10x if x.get_class().value == '0'])

# get cancer (randomized, but reproducible)
cancer_indices = np.array([x.row_index for x in tab_10x if x.get_class().value == '1'])
cancer_indices_randomized = np.random.default_rng(42).permutation(cancer_indices)

# get 1 in 10 cancer
cancer_indices_tenth = cancer_indices_randomized[-int(len(cancer_indices) / 10):]

# create new table with cancer:non-cancer == 1/10:1
tab_10x_downsampl = Orange.data.Table.from_table_rows(tab_10x,
```

```
                                                np.concatenate([cancer_indices_tenth,
                                                                non_cancer_indices]))

tab_10x.name = 'Ming et al. (10x)'
tab_10x_downsampl.name = 'Ming et al. (10x, cancer-downsample)'
```

Get `Race` distribution and cancer occurrence in population sample (10:1 healthy:cancer)

In [34]:
```
# get distribution of race
df_race_signal_10x_dec = pd.DataFrame()
df_race_signal_10x_dec['race'] = tab_10x_downsampl.X[:,-1].astype(int)
df_race_signal_10x_dec_counts = df_race_signal_10x_dec.value_counts().sort_index()
n_signal_race = df_race_signal_10x_dec_counts.values.sum()

# get occurrence of cancer by race
df_race_signal_10x_dec['cancer'] = tab_10x_downsampl.Y.astype(int)
p_cancer_by_race = df_race_signal_10x_dec.groupby('race').mean('cancer')

df_race_signal_10x_downsampl_prop = pd.DataFrame({'count_n_10x_dec': df_race_signal_10x_dec_counts,
                                                  'count_p_10x_dec': df_race_signal_10x_dec_counts / n_signal_race,
                                                  'cancer_10x_dec': p_cancer_by_race.values.flatten()})
table = pd.concat([df_race_signal_10x_prop, df_race_signal_10x_downsampl_prop], axis=1)
table_10x = table.iloc[:, [0,3,2,5,1,4]]
table_10x
```

Out[34]:

race	count_n_10x	count_n_10x_dec	cancer_10x	cancer_10x_dec	count_p_10x	count_p_10x_dec
0	6043	3313	0.505378	0.097797	0.503583	0.501286
1	2397	1337	0.487693	0.081526	0.199750	0.202300
2	2366	1283	0.502959	0.083398	0.197167	0.194129
3	935	510	0.499465	0.082353	0.077917	0.077167
4	126	80	0.436508	0.112500	0.010500	0.012105
5	68	45	0.352941	0.022222	0.005667	0.006809
6	65	41	0.476923	0.170732	0.005417	0.006204

We can see that in the larger population sample has provided what appears to be a viable representation of all races in both the 1:1 and the downsampled 10:1 (healthy:cancer) data.

The most obvious difference being, there are no `race` populations in either sample that have a cancer prevalence of 0%.

Get Cancer Correlation with Independent Variables

In [35]:
```
df_tab_10x = util.table_to_df(tab_10x, x_only=False)
df_cancer_X_correlation_matrix_signal_10x = pd.DataFrame(df_tab_10x.corr().iloc[-1, :]).T

df_tab_10x_downsampl = util.table_to_df(tab_10x_downsampl, x_only=False)
df_cancer_X_correlation_matrix_signal_10x_dec = pd.DataFrame(df_tab_10x_downsampl.corr().iloc[-1, :]).T
```

In [36]:
```
df_cancer_X_correlation_matrix_signal  # 1:1 - healthy:cancer (n=1200)
```

Out[36]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.773642	0.121657	-0.11216	0.061395	0.046615	0.063812	0.025359	1.0

In [37]:
```
df_cancer_X_correlation_matrix_signal_dec  # 10:1 - healthy:cancer (n=1200)
```

Out[37]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.584841	0.106319	-0.088406	0.055188	-0.004393	0.081889	-0.040239	1.0

In [38]:
```
df_cancer_X_correlation_matrix_signal_10x  # 1:1 healthy:cancer (n=12000)
```

Out[38]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.767549	0.081638	-0.073791	0.050976	-0.004405	0.061774	-0.014273	1.0

In [39]:
```
df_cancer_X_correlation_matrix_signal_10x_dec  # 10:1 healthy:cancer (n=12000)
```

Out[39]:

	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	Case_signalYN
Case_signalYN	0.567539	0.047534	-0.033543	0.012155	0.001164	0.046679	-0.014438	1.0

The impact on correlation coefficients is consistent with the prior experiment.

The correlation coefficients, by and large, retain their respective proportions and sign change only occurrs for the single most weakly correlated variable.

This is a reasonable result.

As such, we shall proceed.

Ming et al. Data (10x)

(original n=12000; cancer-downsample n=6609)

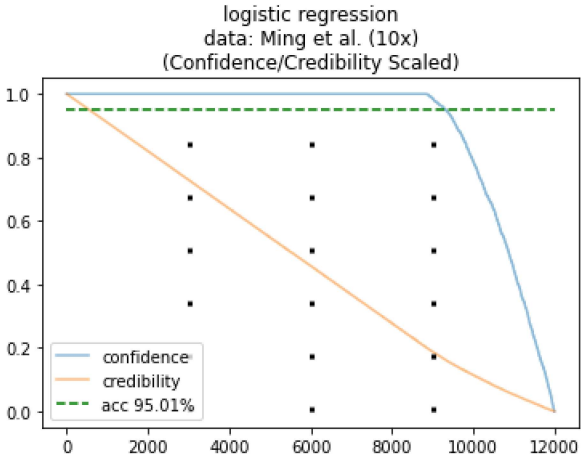
Logistic Regression

In [40]:
```
cc = cp.classification.InductiveClassifier(
        cp.nonconformity.InverseProbability(
            Orange.classification.LogisticRegressionLearner()))

experiment_logistic_signal_10x = util.run_experiments(cc,
                                                      tab_10x,
                                                      tab_10x,
```

```
                                                tab_10x,
                                                eps=0.05,
                                                method='Baseline 10x Unsplit Data')

util.plot_experiments(experiment_logistic_signal_10x)
experiment_logistic_signal_10x.iloc[:,:-3]
```

Out[40]:

	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier	model	conformal_predictor	nonconformity
0	0.991913	0.508163	0.950083	0.0	0.93375	0.883833	0.06625	Ming et al. (10x)	False	logistic regression	LogisticRegressionClassifier(skl_model=Logisti...	InductiveClassifier (InverseProbability (logis...	InverseProbability (logistic regression)



In [41]:

```
cc = cp.classification.InductiveClassifier(
        cp.nonconformity.InverseProbability(
            Orange.classification.LogisticRegressionLearner()))

experiment_logistic_10x_downsampl = util.run_experiments(cc,
                                                         tab_10x_downsampl,
                                                         tab_10x_downsampl,
                                                         tab_10x_downsampl,
                                                         eps=0.05,
                                                         method='Baseline 10x Unsplit Data')

util.plot_experiments(experiment_logistic_10x_downsampl)
experiment_logistic_10x_downsampl.iloc[:,:-3]
```

Out[41]:

	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier	model	conformal_predictor	nonconformity
0	0.99814	0.502021	0.950219	0.036314	0.963686	0.950219	0.0	Ming et al. (10x, cancer-downsample)	False	logistic regression	LogisticRegressionClassifier(skl_model=Logisti...	InductiveClassifier (InverseProbability (logis...	InverseProbability (logistic regression)

Get Lower Decile Range (LDR) Predictions - `confidence` / `credibility`

In [42]:
```
df_pred = experiment_logistic_signal_10x.loc[0, 'df']
df_pred = util.sort_reindex(df_pred, col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_signal_10x_ldr = df_pred.iloc[-ldr_idx:,:].copy()
df_pred_logistic_signal_10x_ldr.tail(1)
```

Out[42]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
11999	[0, 1]	0.92209	0.07791	0.05	[(0.07791017415215398, 0), (0.0779101741521539...	True	-0.293725	2	0	2.407616	-0.416625	0	0	0

In [43]:
```
df_pred = experiment_logistic_10x_downsampl.loc[0, 'df']
df_pred = util.sort_reindex(df_pred, col=['confidence','credibility'])
ldr_idx = int(len(df_pred.index.values) / 10)
df_pred_logistic_10x_downsampl_ldr = df_pred.iloc[-ldr_idx:,:].copy()
df_pred_logistic_10x_downsampl_ldr.tail(1)
```

Out[43]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
6608	[]	0.977458	0.022542	0.05	[(0.02254160363086233, 0), (0.0225416036308623...	False	0.301329	0	2	1.442897	-0.416625	0	0	1

`Race` distribution in LDR

In [44]:
```
# 1:1 healthy:cancer
counts = df_pred_logistic_signal_10x_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_10x_unsplit': counts / sums,
                            'ldr_n_10x_unsplit': counts})
table = table_10x.join(proportions,
                       on='race',
                       how='left')
table = table.fillna(0)
table['ldr_n_10x_unsplit'] = table['ldr_n_10x_unsplit'].astype(int)
```

```
df_logistic_ldr_race_disparity_10x = table.copy()
df_logistic_table_10x = df_logistic_ldr_race_disparity_10x.iloc[:, [0,1,7,2,3,4,5,6]]

# 10:1 healthy:cancer
counts = df_pred_logistic_10x_downsampl_ldr.Race.value_counts().sort_index()
sums = counts.values.sum()
proportions = pd.DataFrame({'ldr_p_10x_unsplit_dec': counts / sums,
                            'ldr_n_10x_unsplit_dec': counts})
table = df_logistic_table_10x.join(proportions,
                                   on='race',
                                   how='left')
table = table.fillna(0)
table['ldr_n_10x_unsplit_dec'] = table['ldr_n_10x_unsplit_dec'].astype(int)
df_logistic_ldr_race_disparity_10x_downsampl = table.copy()
df_logistic_table_10x = df_logistic_ldr_race_disparity_10x_downsampl.iloc[:, [0,1,2,9,3,4,5,6,7,8]]
df_logistic_table_10x
```

Out[44]:

race	count_n_10x	count_n_10x_dec	ldr_n_10x_unsplit	ldr_n_10x_unsplit_dec	cancer_10x	cancer_10x_dec	count_p_10x	count_p_10x_dec	ldr_p_10x_unsplit	ldr_p_10x_unsplit_dec
0	6043	3313	587	353	0.505378	0.097797	0.503583	0.501286	0.489167	0.534848
1	2397	1337	249	121	0.487693	0.081526	0.199750	0.202300	0.207500	0.183333
2	2366	1283	246	107	0.502959	0.083398	0.197167	0.194129	0.205000	0.162121
3	935	510	85	54	0.499465	0.082353	0.077917	0.077167	0.070833	0.081818
4	126	80	17	13	0.436508	0.112500	0.010500	0.012105	0.014167	0.019697
5	68	45	7	3	0.352941	0.022222	0.005667	0.006809	0.005833	0.004545
6	65	41	9	9	0.476923	0.170732	0.005417	0.006204	0.007500	0.013636

---

n=1200

1:1 race proportion sample vs low-confidence --- **10:1** race proportion sample vs low-confidence

In [45]:
```
df_logistic_table.loc[:, 'cancer':].iloc[:, [2,6,3,7]]
```

Out[45]:

race	count_p	ldr_p_unsplit	count_p_dec	ldr_p_dec_unsplit
0	0.490833	0.416667	0.507576	0.590909
1	0.195000	0.275000	0.198485	0.196970
2	0.216667	0.208333	0.200000	0.075758
3	0.078333	0.100000	0.078788	0.121212
4	0.014167	0.000000	0.009091	0.015152
5	0.003333	0.000000	0.003030	0.000000
6	0.001667	0.000000	0.003030	0.000000

---

n=12000

1:1 race proportion sample vs low-confidence --- **10:1** race proportion sample vs low-confidence

In [46]:
```
df_logistic_table_10x.loc[:, 'cancer_10x':].iloc[:, [2,4,3,5]]
```

Out[46]:

race	count_p_10x	ldr_p_10x_unsplit	count_p_10x_dec	ldr_p_10x_unsplit_dec
0	0.503583	0.489167	0.501286	0.534848
1	0.199750	0.207500	0.202300	0.183333
2	0.197167	0.205000	0.194129	0.162121
3	0.077917	0.070833	0.077167	0.081818
4	0.010500	0.014167	0.012105	0.019697
5	0.005667	0.005833	0.006809	0.004545
6	0.005417	0.007500	0.006204	0.013636

Plot population sample race distribution vs low confidence prediction race distribution

We can see that with the data in our original sample of size of 1,200 (Ming et al.) we did not encounter a set of predictions where at least one instance of each race was present in the LDR (low confidence region).

Subsequently, we generated a sample of size 12,000--using the same procedure (Ming et al.)--and performed the (unsplit) 1:1 and 10:1 (health:cancer) experiments with this data.

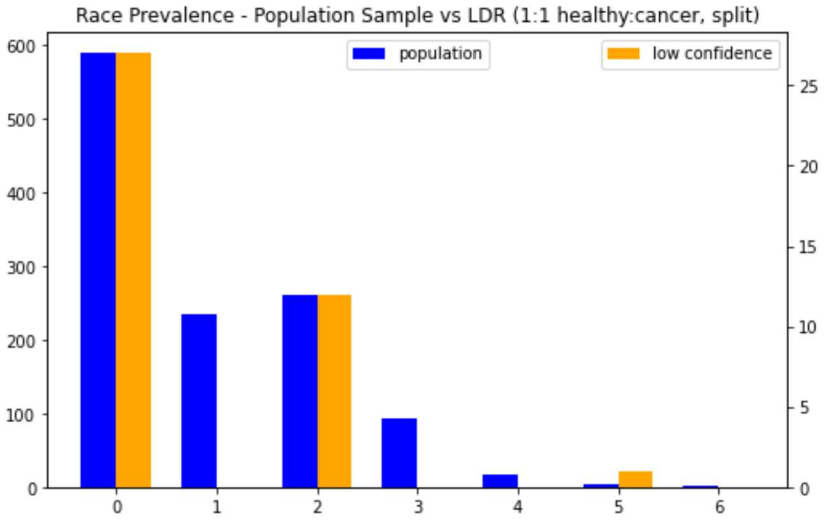
Results n=1200

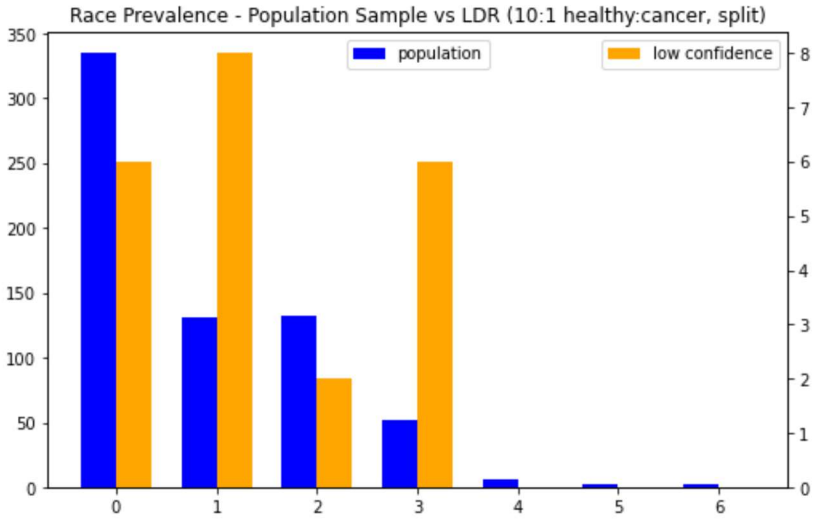
In [47]:
```
fig, axs = plt.subplots(2, 2)

ax = axs[0,0]
x = np.arange(len(df_logistic_table.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table.count_n.sort_index().values
lcr_race = df_logistic_table.ldr_n_split.sort_index().values
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - Population Sample vs LDR (1:1 healthy:cancer, split)')

ax = axs[0,1]
x = np.arange(len(df_logistic_table.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table.count_n_dec.sort_index().values
lcr_race = df_logistic_table.ldr_n_dec_split.sort_index().values
```

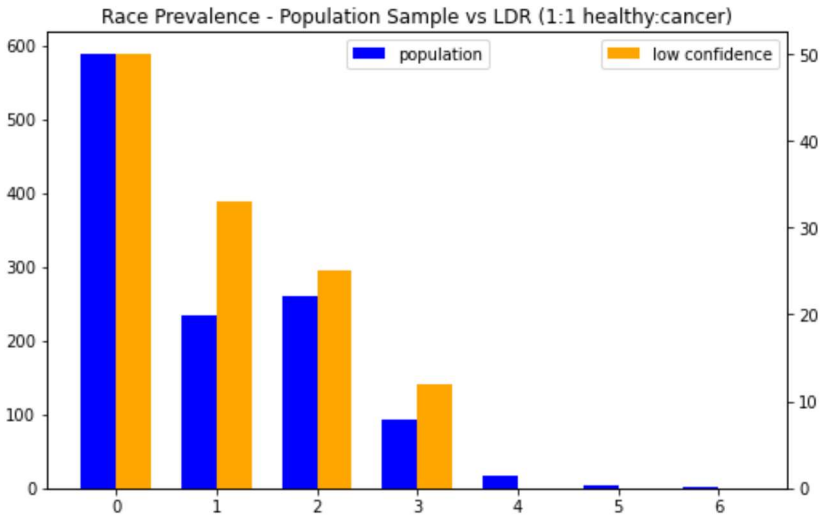
```
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - Population Sample vs LDR (10:1 healthy:cancer, split)')

ax = axs[1,0]
x = np.arange(len(df_logistic_table.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table.count_n.sort_index().values
lcr_race = df_logistic_table.ldr_n_unsplit.sort_index().values
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - Population Sample vs LDR (1:1 healthy:cancer)')

ax = axs[1,1]
x = np.arange(len(df_logistic_table.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table.count_n_dec.sort_index().values
lcr_race = df_logistic_table.ldr_n_dec_unsplit.sort_index().values
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - Population Sample vs LDR (10:1 healthy:cancer)')

fig.set_figwidth(18)
fig.set_figheight(12)
df_logistic_table
```

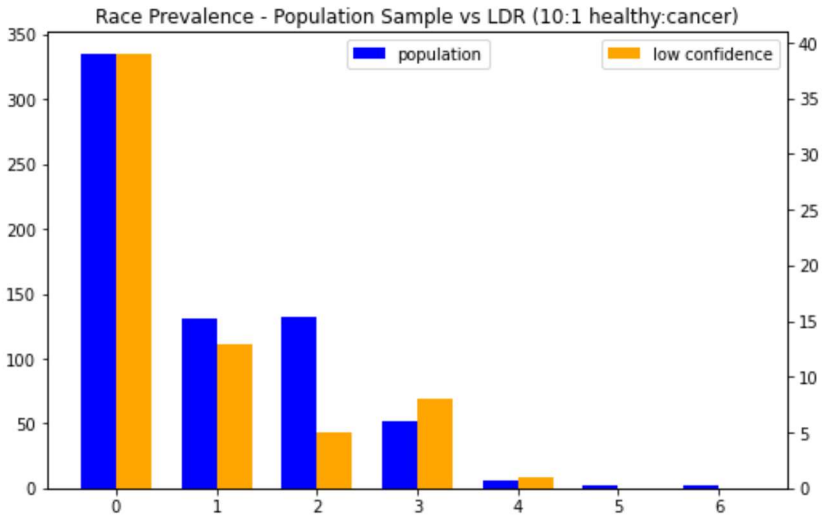
Out[47]:

race	count_n	count_n_dec	ldr_n_split	ldr_n_dec_split	ldr_n_unsplit	ldr_n_dec_unsplit	cancer	cancer_dec	count_p	count_p_dec	ldr_p_split	ldr_p_dec_split	ldr_p_unsplit	ldr_p_dec_unsplit
0	589	335	27	6	50	39	0.490662	0.104478	0.490833	0.507576	0.675	0.272727	0.416667	0.590909
1	234	131	0	8	33	13	0.478632	0.068702	0.195000	0.198485	0.000	0.363636	0.275000	0.196970
2	260	132	12	2	25	5	0.526923	0.068182	0.216667	0.200000	0.300	0.090909	0.208333	0.075758
3	94	52	0	6	12	8	0.510638	0.115385	0.078333	0.078788	0.000	0.272727	0.100000	0.121212
4	17	6	0	0	0	1	0.647059	0.000000	0.014167	0.009091	0.000	0.000000	0.000000	0.015152
5	4	2	1	0	0	0	0.500000	0.000000	0.003333	0.003030	0.025	0.000000	0.000000	0.000000
6	2	2	0	0	0	0	0.000000	0.000000	0.001667	0.003030	0.000	0.000000	0.000000	0.000000









We can clearly see from the upper graphs that the results of the split experiments are highly variable compared to their lower, unsplit counterparts.

This is, in large part, likely due to the small quantity of individuals belonging to the three least prevalent races.

In the 1:1 split/unsplit experiments, these three races total 23 individuals.

In the corresponding 10:1 split/unsplit experiments, these three races total 10 individuals. In spite of this, `race` 4 has (possibly appropriate) representation in the LDR of the unsplit experiment.

For these reasons, we **remove the bias introduced by splitting the data** and utilize a dataset we can more reasonably expect to provide a **viable representation of the variance in the minority `race` populations**.

This is the *modus operandi* of the experiments for which we present the results of below

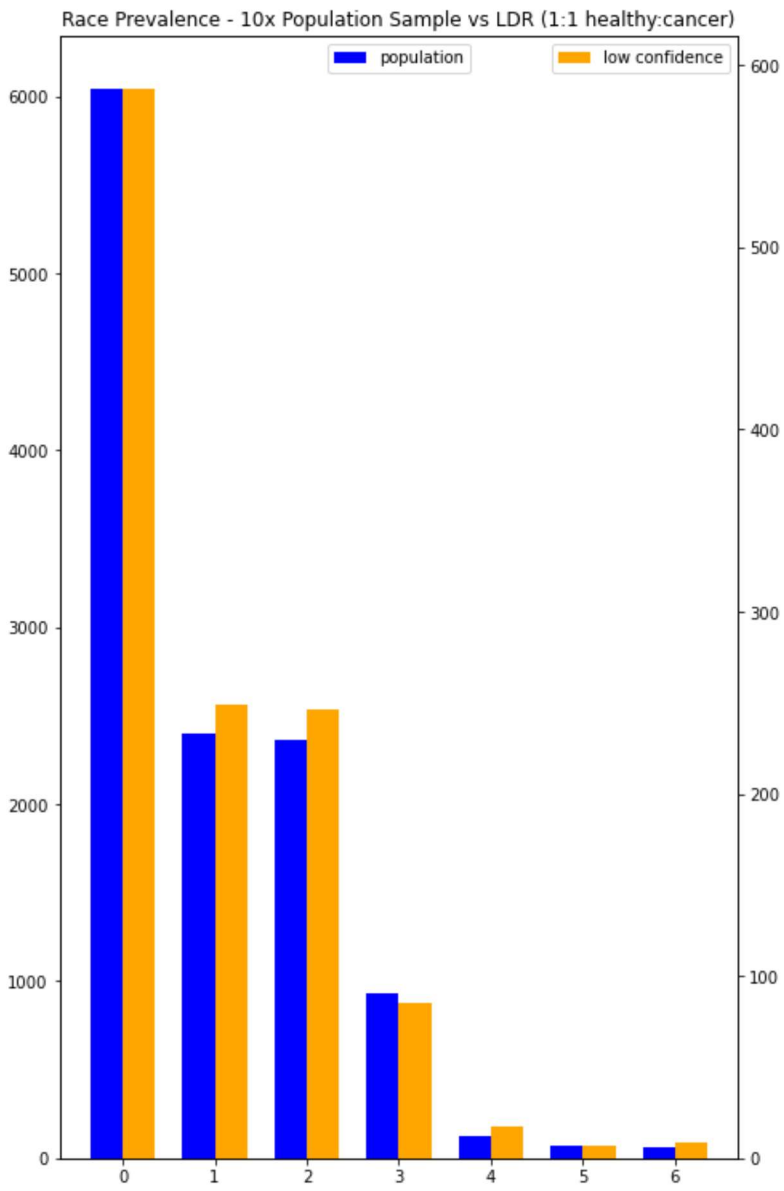
---

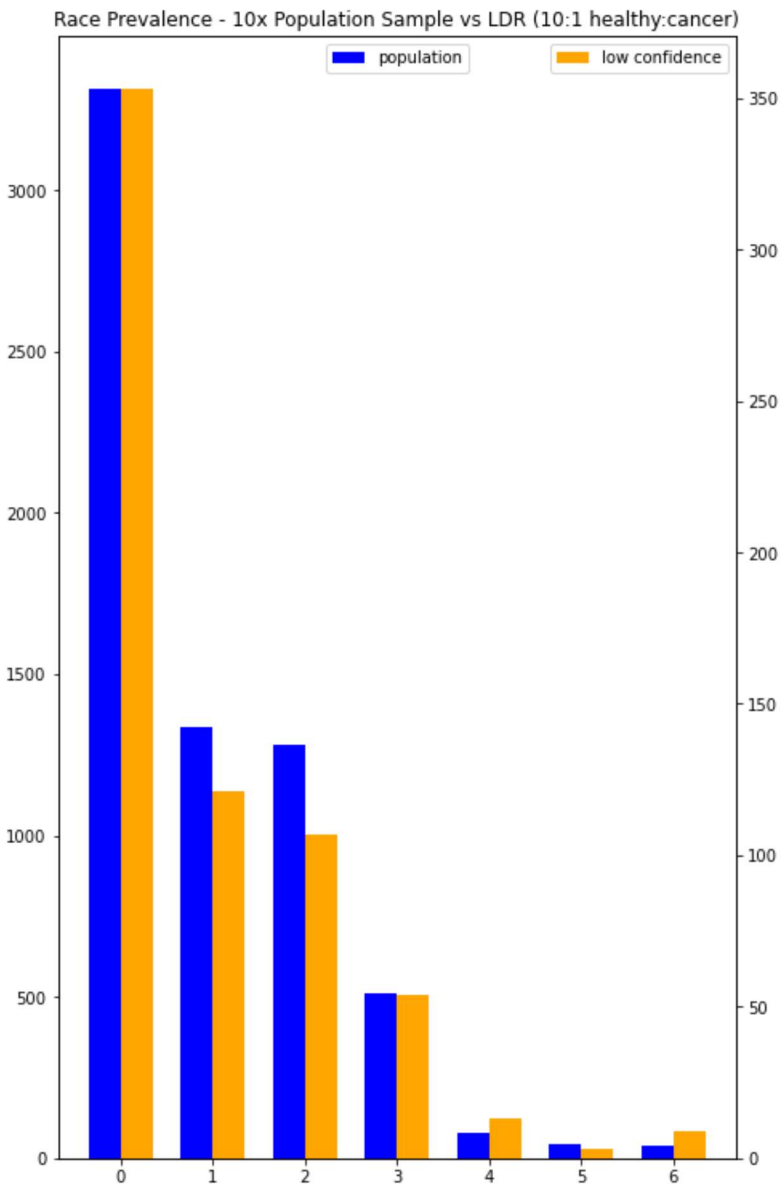
Results n=12000

In [48]:
```
fig, axs = plt.subplots(1, 2)

ax = axs[0]
x = np.arange(len(df_logistic_table_10x.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table_10x.count_n_10x.sort_index().values
lcr_race = df_logistic_table_10x.ldr_n_10x_unsplit.sort_index().values
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - 10x Population Sample vs LDR (1:1 healthy:cancer)')

ax = axs[1]
x = np.arange(len(df_logistic_table_10x.index))
ax_y2 = ax.twinx()
width = 0.35
pop_race = df_logistic_table_10x.count_n_10x_dec.sort_index().values
lcr_race = df_logistic_table_10x.ldr_n_10x_unsplit_dec.sort_index().values
ax.bar(x - width/2, pop_race, width, color='blue', label='population')
ax_y2.bar(x + width/2, lcr_race, width, color='orange', label='low confidence')
ax.legend(loc='upper center')
ax_y2.legend()
fig.tight_layout()
ax.set_title('Race Prevalence - 10x Population Sample vs LDR (10:1 healthy:cancer)')

fig.set_figwidth(18)
fig.set_figheight(12)

df_logistic_table_10x
```

Out[48]:

race	count_n_10x	count_n_10x_dec	ldr_n_10x_unsplit	ldr_n_10x_unsplit_dec	cancer_10x	cancer_10x_dec	count_p_10x	count_p_10x_dec	ldr_p_10x_unsplit	ldr_p_10x_unsplit_dec
0	6043	3313	587	353	0.505378	0.097797	0.503583	0.501286	0.489167	0.534848
1	2397	1337	249	121	0.487693	0.081526	0.199750	0.202300	0.207500	0.183333
2	2366	1283	246	107	0.502959	0.083398	0.197167	0.194129	0.205000	0.162121
3	935	510	85	54	0.499465	0.082353	0.077917	0.077167	0.070833	0.081818
4	126	80	17	13	0.436508	0.112500	0.010500	0.012105	0.014167	0.019697
5	68	45	7	3	0.352941	0.022222	0.005667	0.006809	0.005833	0.004545
6	65	41	9	9	0.476923	0.170732	0.005417	0.006204	0.007500	0.013636





We can see that with the data in our sample of size of 12,000 (Ming et al.) we encounter at least one instance of each race in the LDR (low confidence region) for both (1:1 & 10:1) experiments.

Additionally, we observe that in both experiments:

- **Majority** `race` 0 (**~50%** of the population sample)
  - **Does not** appear to tend toward **over-representation** in the **low confidence region** of either experiments' predictions
- **Minority** `race` 4, 5, and 6 (each representing **< 1.25%** of the sample)
  - **Does** appear to tend toward **over-represention** in the **low confidence region** of both experiments' predictions
    - The exception is `race` 5 in the 10:1 experiment (0.58% vs 0.45%)
      - However, the small quantity of individuals belonging to this race (n=68) and the prevalence of cancer in individuals belonging to this race (1/68) may help account for the observed lack of low confidence predictions

---

Contents

Visualization of results from `LogisticRegression`, `RandomForest`, `KNeighborsClassifier`, and `AdaBoostClassifier` experiments using the Nonconformity Measure `InverseProbability`.

Distance-based `KNNFraction` Nonconformity Measure was applied to the the same experiment setups as a reference point.

`KNNFraction` uses `Euclidean`, similar to `KNeighborsClassifier`, and is not applied to the underlying classifiers above.

In [60]:
```
# 'vigilant-computing-machine/source/experiment.py'
import source.experiment as exp
```

Experiments:

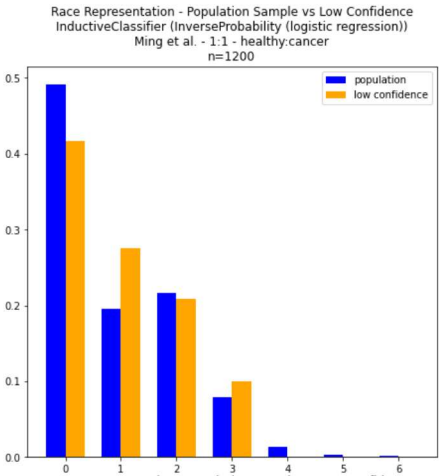
- `n=1200`
  - **Mondrian** = **False**
    - **Balanced** = **1:1** healthy:cancer
    - **Imbalanced** = **10:1** healthy:cancer
  - **Mondrian** = **True**
    - **Balanced** = **1:1** healthy:cancer
    - **Imbalanced** = **10:1** healthy:cancer
- `n=12000`
  - **Mondrian** = **False**
    - **Balanced** = **1:1** healthy:cancer
    - **Imbalanced** = **10:1** healthy:cancer
  - **Mondrian** = **True**
    - **Balanced** = **1:1** healthy:cancer
    - **Imbalanced** = **10:1** healthy:cancer

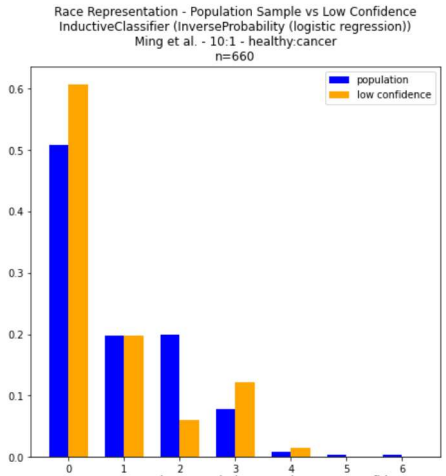
`n=1200` Experiments

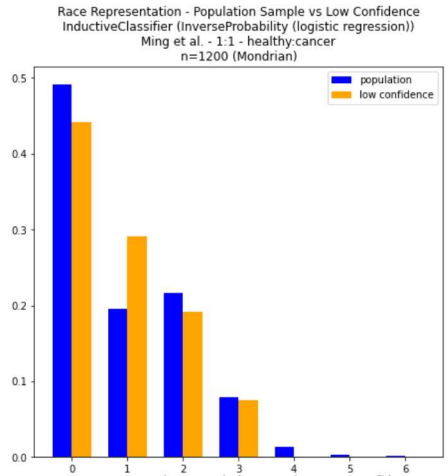
Read in and Plot Experiment Results

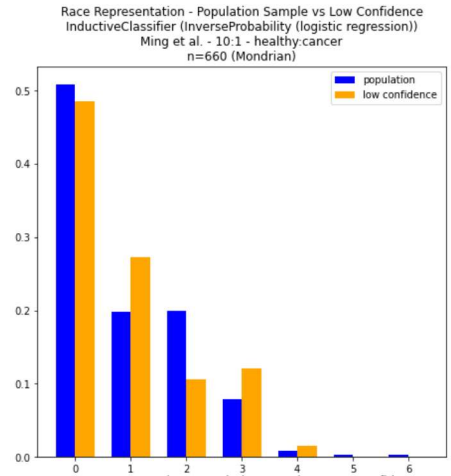
In [61]:

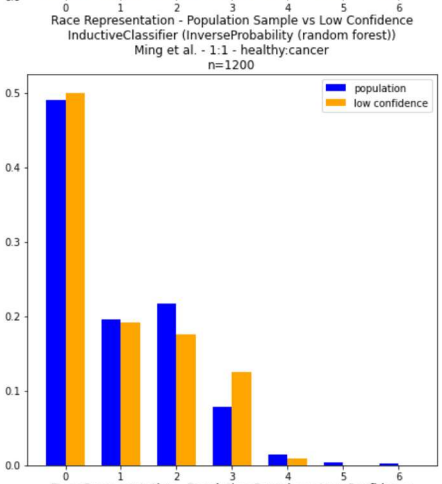
```
experiments = [util.read_experiment('./results/logistic_regression_1200_balanced_experiment.csv'),
               util.read_experiment('./results/logistic_regression_1200_imbalanced_experiment.csv'),
               util.read_experiment('./results/logistic_regression_1200_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/logistic_regression_1200_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/rf_1200_balanced_experiment.csv'),
               util.read_experiment('./results/rf_1200_imbalanced_experiment.csv'),
               util.read_experiment('./results/rf_1200_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/rf_1200_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/knn_1200_balanced_experiment.csv'),
               util.read_experiment('./results/knn_1200_imbalanced_experiment.csv'),
               util.read_experiment('./results/knn_1200_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/knn_1200_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/ada_1200_balanced_experiment.csv'),
               util.read_experiment('./results/ada_1200_imbalanced_experiment.csv'),
               util.read_experiment('./results/ada_1200_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/ada_1200_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/knn_fraction_1200_balanced_experiment.csv'),
               util.read_experiment('./results/knn_fraction_1200_imbalanced_experiment.csv'),
               util.read_experiment('./results/knn_fraction_1200_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/knn_fraction_1200_imbalanced_mondrian_experiment.csv')]


col = 4
row = 5

fig, axs = plt.subplots(row, col)
fig.set_figheight(42)
fig.set_figwidth(34)

for i in range(row * col):
    row_idx = i // col
    col_idx = i % col
    util.plot_race_representation_from_experiment(experiments[i], ax=axs[row_idx, col_idx])
```

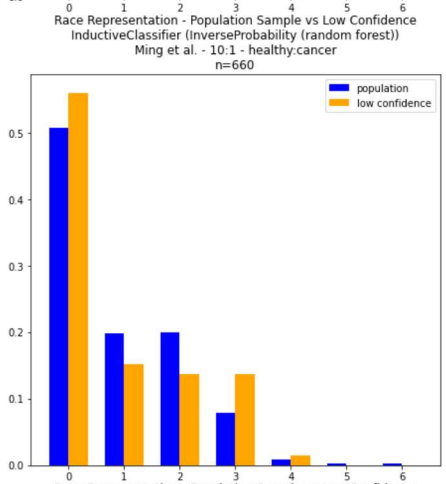

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
Ming et al. - 1:1 - healthy:cancer
n=1200
population
low confidence

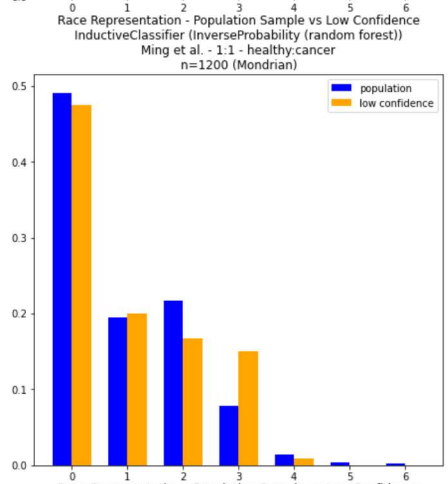

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
Ming et al. - 10:1 - healthy:cancer
n=660
population
low confidence

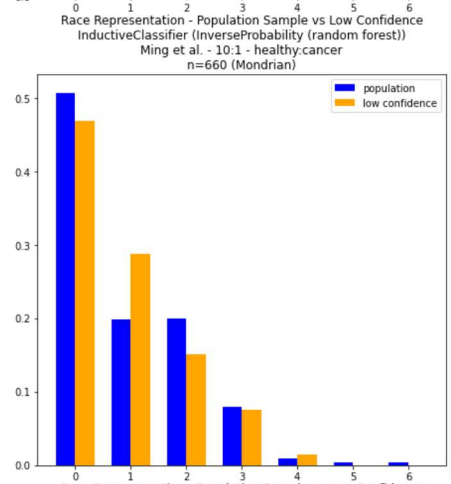

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
Ming et al. - 1:1 - healthy:cancer
n=1200 (Mondrian)
population
low confidence

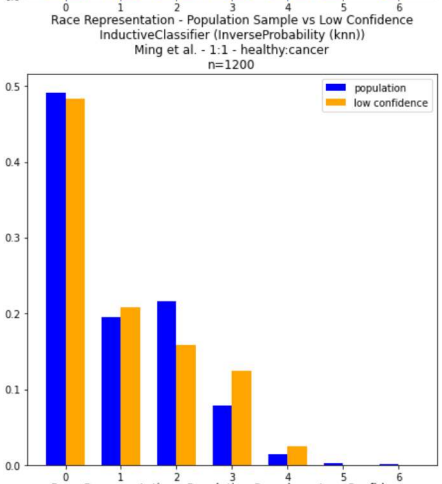

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
Ming et al. - 10:1 - healthy:cancer
n=660 (Mondrian)
population
low confidence

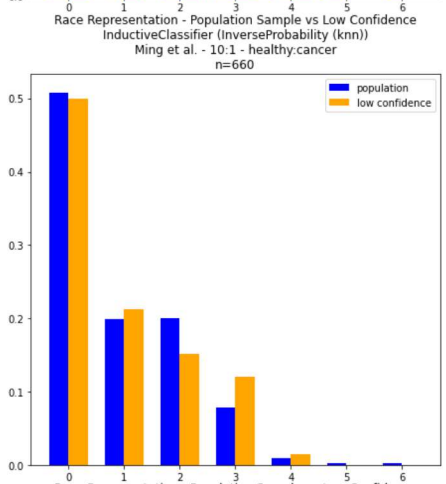

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
Ming et al. - 1:1 - healthy:cancer
n=1200
population
low confidence

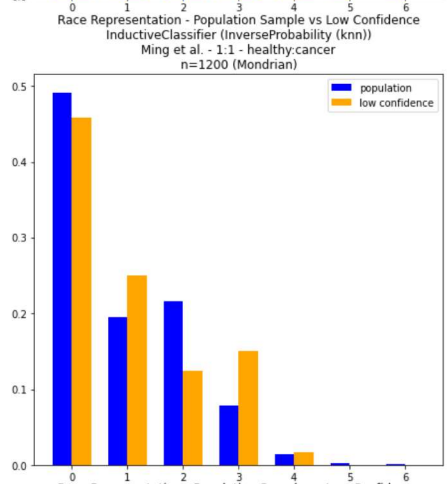

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
Ming et al. - 10:1 - healthy:cancer
n=660
population
low confidence

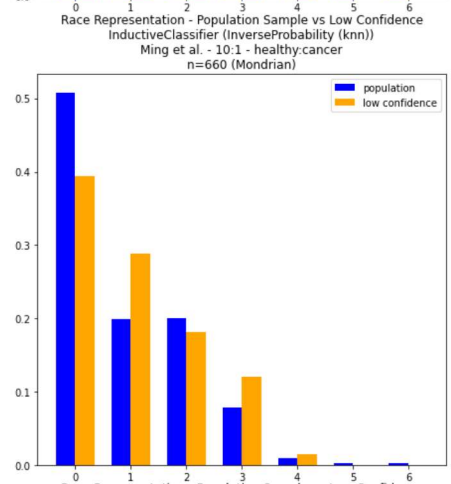

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
Ming et al. - 1:1 - healthy:cancer
n=1200 (Mondrian)
population
low confidence

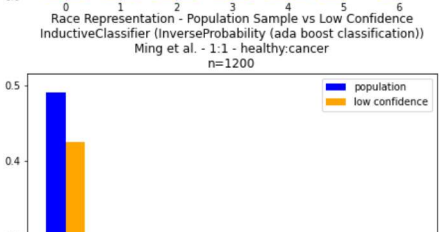

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
Ming et al. - 10:1 - healthy:cancer
n=660 (Mondrian)
population
low confidence

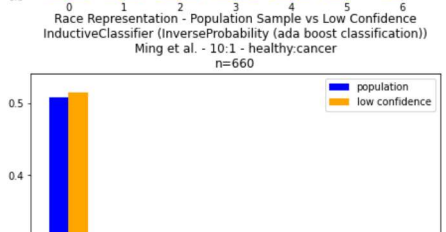

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
Ming et al. - 1:1 - healthy:cancer
n=1200
population
low confidence

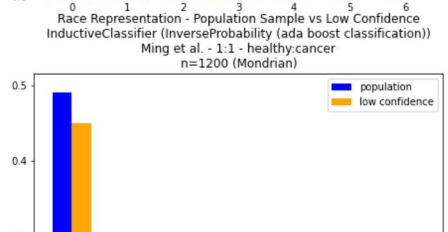

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
Ming et al. - 10:1 - healthy:cancer
n=660
population
low confidence

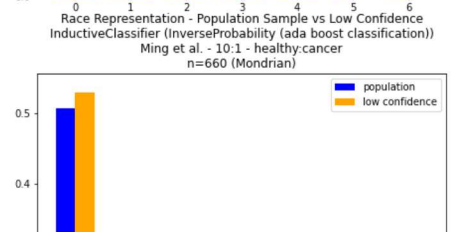

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
Ming et al. - 1:1 - healthy:cancer
n=1200 (Mondrian)
population
low confidence

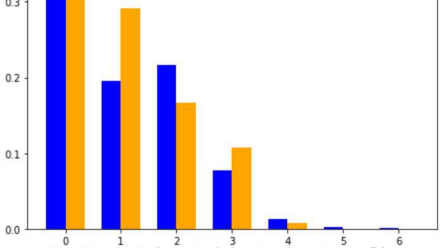

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
Ming et al. - 10:1 - healthy:cancer
n=660 (Mondrian)
population
low confidence

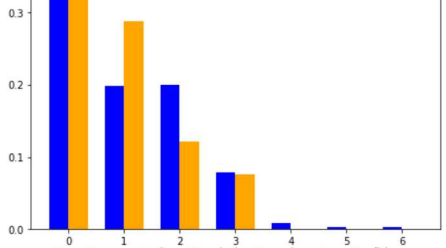

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
Ming et al. - 1:1 - healthy:cancer
n=1200
population
low confidence

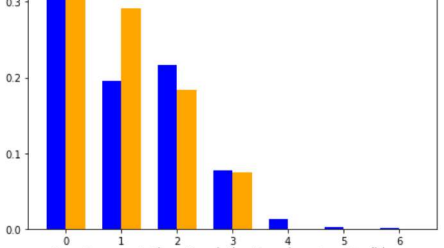

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
Ming et al. - 10:1 - healthy:cancer
n=660
population
low confidence

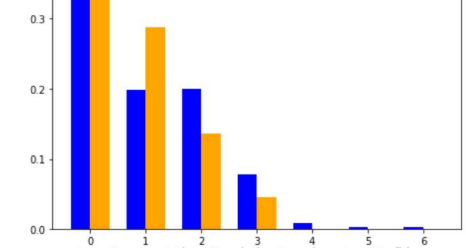

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
Ming et al. - 1:1 - healthy:cancer
n=1200 (Mondrian)
population
low confidence

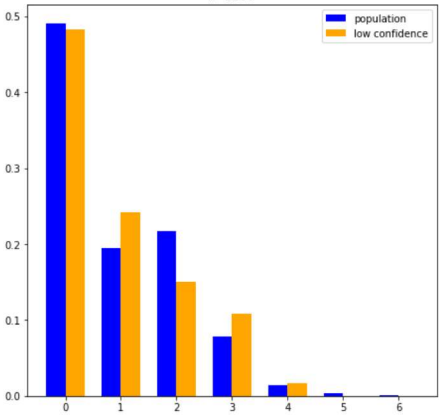

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
Ming et al. - 10:1 - healthy:cancer
n=660 (Mondrian)
population
low confidence

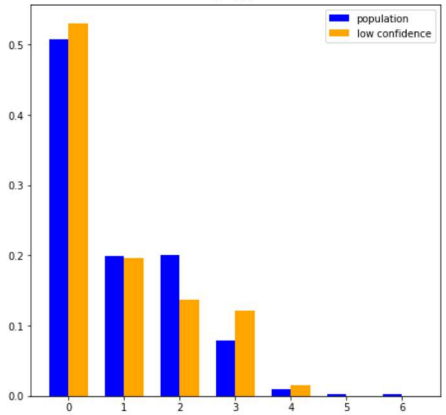


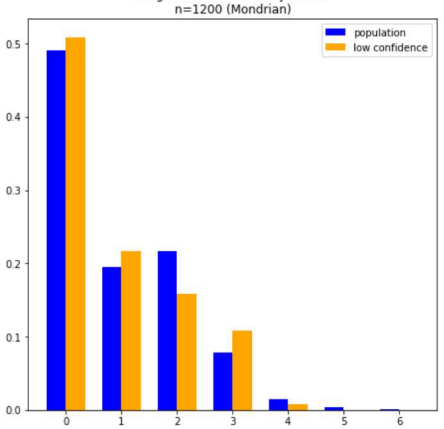


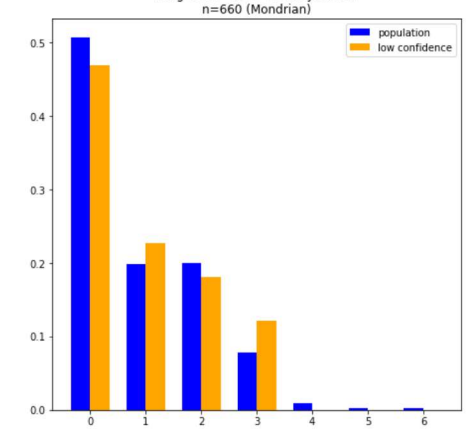












Observation:

Bear in mind that these are the results of experiments utilizing the population sample previously determined as likely to insufficiently represent the most sparsely populated `race` groups.

We can see that in the presense of `race` 4 individuals' predictions in the low confidence region varies among the experiments.

If investigation of `race` 4 indvidiuals indicates that confidence should be low for their predictions, then KNN appears to have been the most well suited underlying classifier to correctly identify this particular situation.

If the opposite is true, then across underlying classifiers, Mondrian Inductive Conformal Predictors more often picked up on this situation than their Inductive Conformal Predictor counterparts.

Note: the full set of predictions are available in each experiments' `'df'` column (e.g., `experiments[0].loc[0, 'df']`)

In [62]:
```
pd.concat(experiments).loc[:, 'confidence':'classifier']
```

Out[62]:

	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier
0	0.993566	0.506468	0.900833	0.069167	0.930833	0.900833	0.000000	Ming et al. - 1:1 - healthy:cancer	False	logistic regression
0	0.997456	0.502686	0.901515	0.093939	0.906061	0.901515	0.000000	Ming et al. - 10:1 - healthy:cancer	False	logistic regression
0	0.992670	0.507476	0.901667	0.066667	0.933333	0.901667	0.000000	Ming et al. - 1:1 - healthy:cancer	True	logistic regression
0	0.982278	0.507575	0.903030	0.063636	0.936364	0.903030	0.000000	Ming et al. - 10:1 - healthy:cancer	True	logistic regression
0	0.998612	0.824756	0.900833	0.099167	0.900833	0.900833	0.000000	Ming et al. - 1:1 - healthy:cancer	False	random forest
0	0.998446	0.884665	0.954545	0.045455	0.954545	0.954545	0.000000	Ming et al. - 10:1 - healthy:cancer	False	random forest
0	0.997761	0.824989	0.917500	0.082500	0.917500	0.917500	0.000000	Ming et al. - 1:1 - healthy:cancer	True	random forest
0	0.984962	0.911020	0.933333	0.066667	0.933333	0.933333	0.000000	Ming et al. - 10:1 - healthy:cancer	True	random forest
0	0.983689	0.734906	0.924167	0.000000	1.000000	0.924167	0.000000	Ming et al. - 1:1 - healthy:cancer	False	knn
0	0.995936	0.894487	0.940909	0.045455	0.954545	0.940909	0.000000	Ming et al. - 10:1 - healthy:cancer	False	knn
0	0.982681	0.735245	0.924167	0.000000	1.000000	0.924167	0.000000	Ming et al. - 1:1 - healthy:cancer	True	knn
0	0.982421	0.925955	0.951515	0.000000	1.000000	0.951515	0.000000	Ming et al. - 10:1 - healthy:cancer	True	knn
0	0.994586	0.505934	0.900833	0.072500	0.927500	0.900833	0.000000	Ming et al. - 1:1 - healthy:cancer	False	ada boost classification
0	0.998164	0.503397	0.901515	0.098485	0.901515	0.901515	0.000000	Ming et al. - 10:1 - healthy:cancer	False	ada boost classification
0	0.993673	0.507443	0.901667	0.071667	0.928333	0.901667	0.000000	Ming et al. - 1:1 - healthy:cancer	True	ada boost classification
0	0.984659	0.507126	0.903030	0.084848	0.915152	0.903030	0.000000	Ming et al. - 10:1 - healthy:cancer	True	ada boost classification
0	0.965999	0.737750	0.958333	0.000000	0.824167	0.782500	0.175833	Ming et al. - 1:1 - healthy:cancer	False	KNN - EuclideanRowsModel
0	0.984161	0.899796	0.937879	0.034848	0.965152	0.937879	0.000000	Ming et al. - 10:1 - healthy:cancer	False	KNN - EuclideanRowsModel
0	0.964554	0.738649	0.926667	0.000000	0.918333	0.845000	0.081667	Ming et al. - 1:1 - healthy:cancer	True	KNN - EuclideanRowsModel
0	0.867723	0.933936	0.939394	0.000000	0.137879	0.077273	0.862121	Ming et al. - 10:1 - healthy:cancer	True	KNN - EuclideanRowsModel

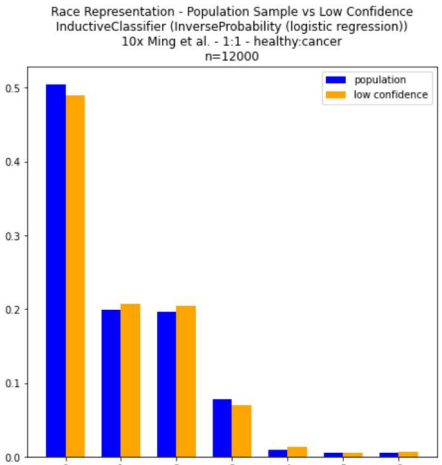
---

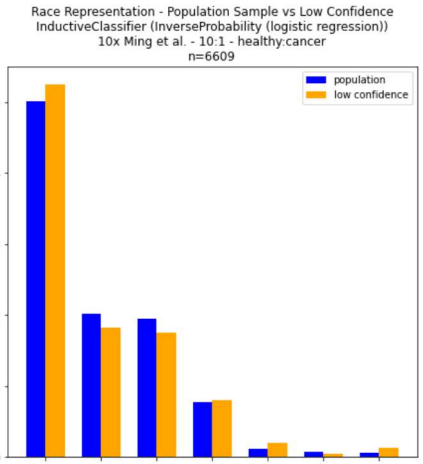
`n=12000` Experiments

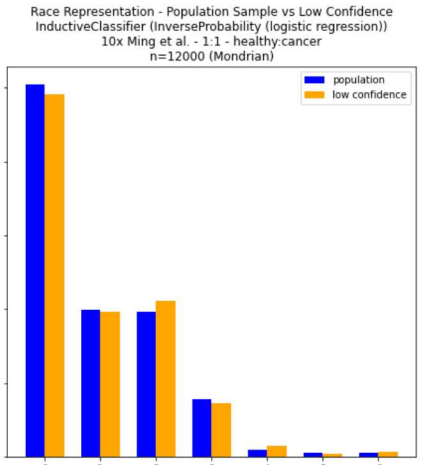
Read in and Plot Experiment Results

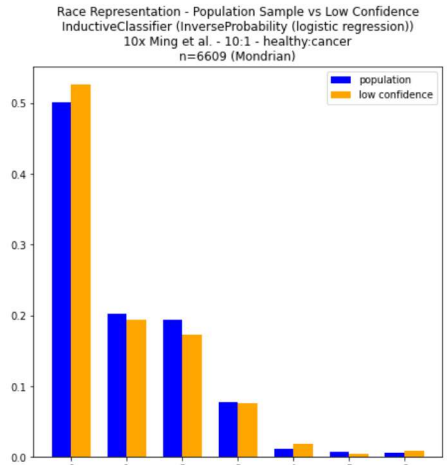
In [63]:
```
experiments = [util.read_experiment('./results/logistic_regression_12000_balanced_experiment.csv'),
               util.read_experiment('./results/logistic_regression_12000_imbalanced_experiment.csv'),
               util.read_experiment('./results/logistic_regression_12000_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/logistic_regression_12000_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/rf_12000_balanced_experiment.csv'),
               util.read_experiment('./results/rf_12000_imbalanced_experiment.csv'),
               util.read_experiment('./results/rf_12000_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/rf_12000_imbalanced_mondrian_experiment.csv'),

               util.read_experiment('./results/knn_12000_balanced_experiment.csv'),
               util.read_experiment('./results/knn_12000_imbalanced_experiment.csv'),
               util.read_experiment('./results/knn_12000_balanced_mondrian_experiment.csv'),
               util.read_experiment('./results/knn_12000_imbalanced_mondrian_experiment.csv'),
```

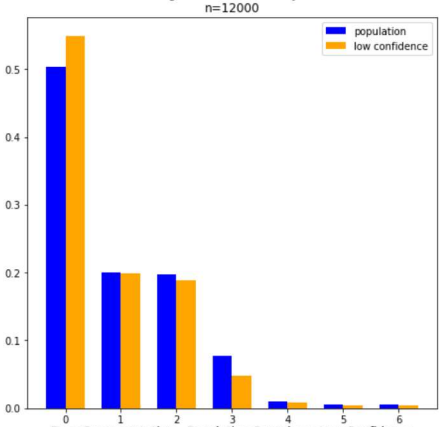
```
                util.read_experiment('./results/ada_12000_balanced_experiment.csv'),
                util.read_experiment('./results/ada_12000_imbalanced_experiment.csv'),
                util.read_experiment('./results/ada_12000_balanced_mondrian_experiment.csv'),
                util.read_experiment('./results/ada_12000_imbalanced_mondrian_experiment.csv'),

                util.read_experiment('./results/knn_fraction_12000_balanced_experiment.csv'),
                util.read_experiment('./results/knn_fraction_12000_imbalanced_experiment.csv'),
                util.read_experiment('./results/knn_fraction_12000_balanced_mondrian_experiment.csv'),
                util.read_experiment('./results/knn_fraction_12000_imbalanced_mondrian_experiment.csv')]


col = 4
row = 5

fig, axs = plt.subplots(row, col)
fig.set_figheight(42)
fig.set_figwidth(34)

for i in range(row * col):
    row_idx = i // col
    col_idx = i % col
    util.plot_race_representation_from_experiment(experiments[i], ax=axs[row_idx, col_idx])
```

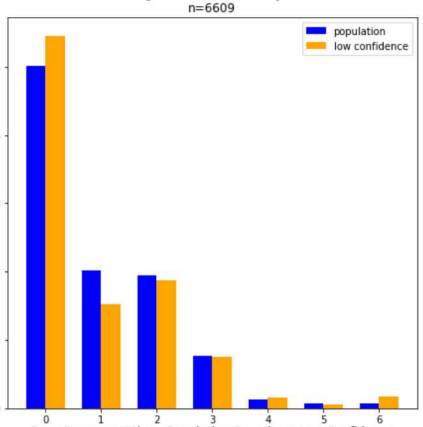

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
10x Ming et al. - 1:1 - healthy:cancer
n=12000
population
low confidence

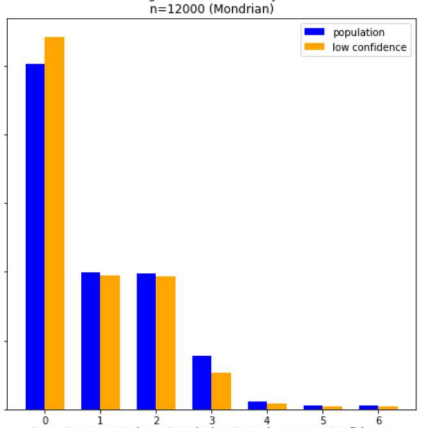

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
10x Ming et al. - 10:1 - healthy:cancer
n=6609
population
low confidence

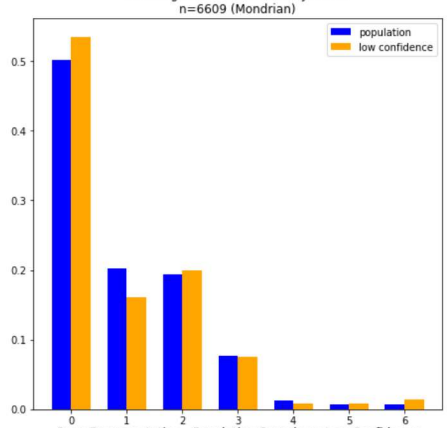

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
10x Ming et al. - 1:1 - healthy:cancer
n=12000 (Mondrian)
population
low confidence

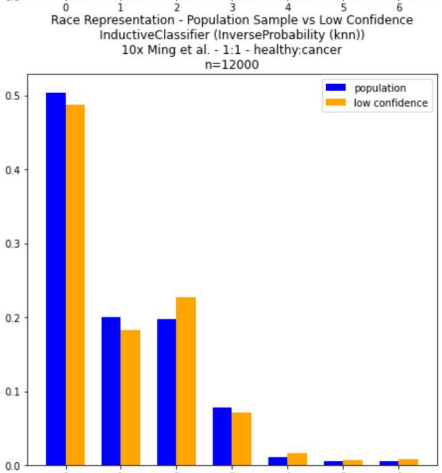

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (logistic regression))
10x Ming et al. - 10:1 - healthy:cancer
n=6609 (Mondrian)
population
low confidence

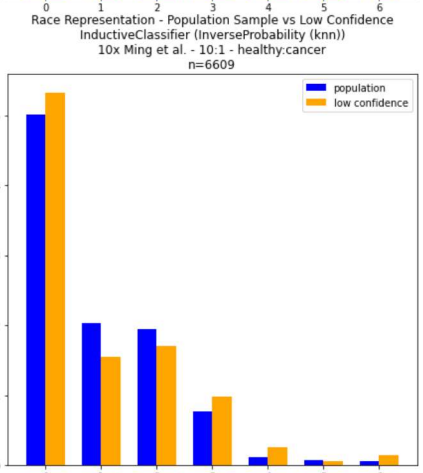

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
10x Ming et al. - 1:1 - healthy:cancer
n=12000
population
low confidence

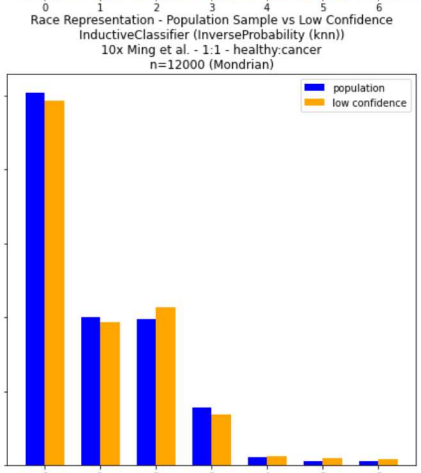

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
10x Ming et al. - 10:1 - healthy:cancer
n=6609
population
low confidence

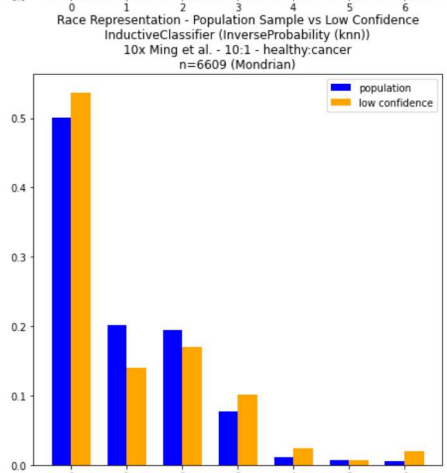

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
10x Ming et al. - 1:1 - healthy:cancer
n=12000 (Mondrian)
population
low confidence

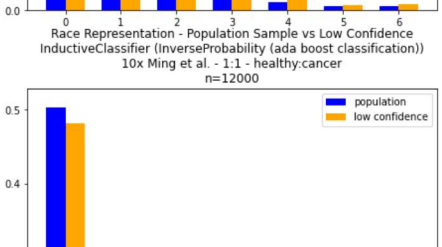

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (random forest))
10x Ming et al. - 10:1 - healthy:cancer
n=6609 (Mondrian)
population
low confidence

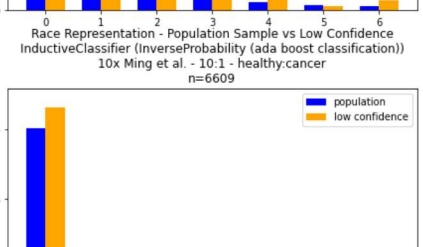

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
10x Ming et al. - 1:1 - healthy:cancer
n=12000
population
low confidence

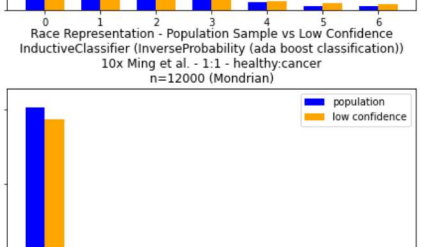

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
10x Ming et al. - 10:1 - healthy:cancer
n=6609
population
low confidence

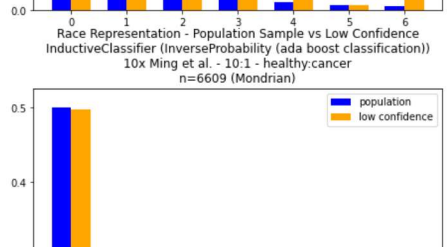

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
10x Ming et al. - 1:1 - healthy:cancer
n=12000 (Mondrian)
population
low confidence

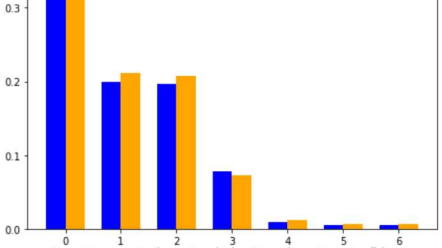

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (knn))
10x Ming et al. - 10:1 - healthy:cancer
n=6609 (Mondrian)
population
low confidence

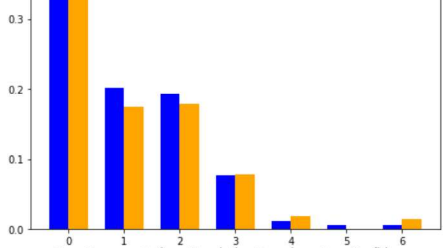

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
10x Ming et al. - 1:1 - healthy:cancer
n=12000
population
low confidence

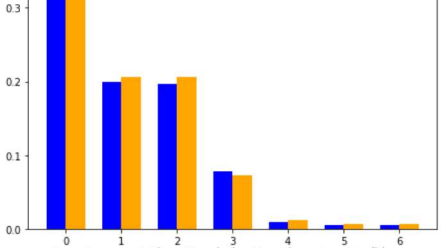

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
10x Ming et al. - 10:1 - healthy:cancer
n=6609
population
low confidence

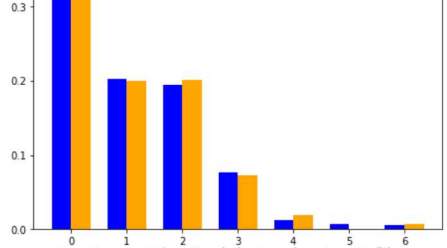

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
10x Ming et al. - 1:1 - healthy:cancer
n=12000 (Mondrian)
population
low confidence

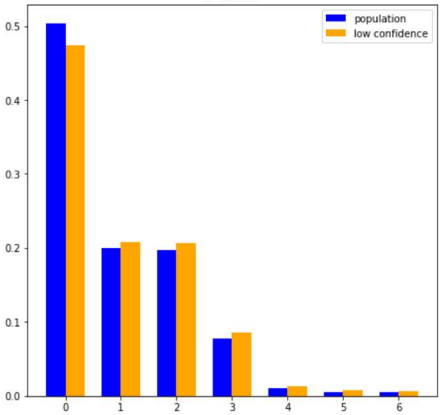

Race Representation - Population Sample vs Low Confidence
InductiveClassifier (InverseProbability (ada boost classification))
10x Ming et al. - 10:1 - healthy:cancer
n=6609 (Mondrian)
population
low confidence

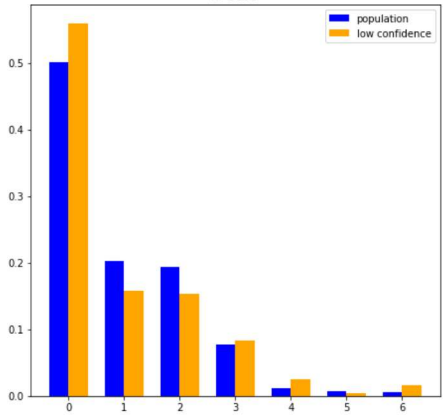


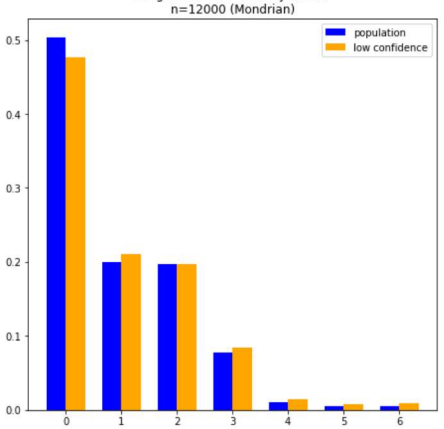


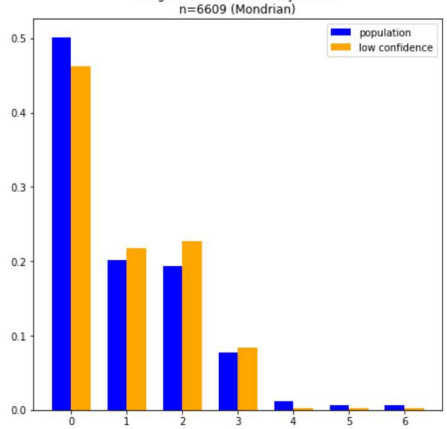












In [64]:
```
pd.concat(experiments).loc[:, 'confidence':'classifier']
```

Out[64]:

	confidence	credibility	verdict	empty	single	single_correct	multiple	data	mondrian	classifier
0	0.991913	0.508163	0.900083	0.039500	0.960500	0.900083	0.000000	10x Ming et al. - 1:1 - healthy:cancer	False	logistic regression
0	0.998153	0.502008	0.901195	0.091844	0.908156	0.901195	0.000000	10x Ming et al. - 10:1 - healthy:cancer	False	logistic regression
0	0.991823	0.508341	0.900333	0.038500	0.961500	0.900333	0.000000	10x Ming et al. - 1:1 - healthy:cancer	True	logistic regression
0	0.990868	0.507395	0.900741	0.050235	0.949765	0.900741	0.000000	10x Ming et al. - 10:1 - healthy:cancer	True	logistic regression
0	0.997174	0.803856	0.908250	0.078750	0.921250	0.908250	0.000000	10x Ming et al. - 1:1 - healthy:cancer	False	random forest
0	0.999604	0.910560	0.905281	0.094719	0.905281	0.905281	0.000000	10x Ming et al. - 10:1 - healthy:cancer	False	random forest
0	0.997082	0.803915	0.901750	0.085750	0.914250	0.901750	0.000000	10x Ming et al. - 1:1 - healthy:cancer	True	random forest
0	0.997808	0.915207	0.929339	0.069148	0.930852	0.929339	0.000000	10x Ming et al. - 10:1 - healthy:cancer	True	random forest
0	0.986283	0.789914	0.929417	0.000000	1.000000	0.929417	0.000000	10x Ming et al. - 1:1 - healthy:cancer	False	knn
0	0.997482	0.885931	0.942805	0.044939	0.955061	0.942805	0.000000	10x Ming et al. - 10:1 - healthy:cancer	False	knn
0	0.986190	0.789951	0.929417	0.000000	1.000000	0.929417	0.000000	10x Ming et al. - 1:1 - healthy:cancer	True	knn
0	0.992075	0.904505	0.923892	0.000000	1.000000	0.923892	0.000000	10x Ming et al. - 10:1 - healthy:cancer	True	knn
0	0.991765	0.508793	0.900083	0.041750	0.958250	0.900083	0.000000	10x Ming et al. - 1:1 - healthy:cancer	False	ada boost classification
0	0.998361	0.503026	0.900287	0.093509	0.906491	0.900287	0.000000	10x Ming et al. - 10:1 - healthy:cancer	False	ada boost classification
0	0.991676	0.509453	0.900167	0.041250	0.958750	0.900167	0.000000	10x Ming et al. - 1:1 - healthy:cancer	True	ada boost classification
0	0.991927	0.508996	0.900439	0.047057	0.952943	0.900439	0.000000	10x Ming et al. - 10:1 - healthy:cancer	True	ada boost classification
0	0.971893	0.788605	0.901667	0.000000	1.000000	0.901667	0.000000	Ming et al. - 1:1 - healthy:cancer	False	KNN - EuclideanRowsModel
0	0.988263	0.886205	0.938720	0.039946	0.960054	0.938720	0.000000	Ming et al. - 10:1 - healthy:cancer	False	KNN - EuclideanRowsModel
0	0.971765	0.788646	0.928333	0.000000	0.934333	0.862667	0.065667	Ming et al. - 1:1 - healthy:cancer	True	KNN - EuclideanRowsModel
0	0.906020	0.909459	0.910879	0.000000	0.170525	0.081404	0.829475	Ming et al. - 10:1 - healthy:cancer	True	KNN - EuclideanRowsModel

---

Visually Representing Individual Prediction

In [117…

```
import plotly.graph_objects as go

def plot_prediction_uncertainty(s):
    pred = s
    min_cred = 5
    min_conf = 90
    conf = pred.confidence*100
    cred = pred.credibility*100
```

```
    second_most_cred = (1 - pred.credibility) * 100
    cred_threshold_clr = 'red' if cred < min_cred else 'lime'
    conf_color = 'red' if conf < min_conf else 'green'

    fig =go.Figure(go.Indicator(
        domain={'x': [0, 1], 'y': [0, 1]},
        value=conf,
        mode='gauge+number+delta',
        title={'text':'Certainty'},
        delta={'reference':second_most_cred},
        gauge={'axis': {'range': [None,100]},
               'steps': [
                   {'range': [0,min_cred], 'color': 'tomato'},
                   {'range': [min_conf,100], 'color': 'lightgray'}],
               'threshold': {'line': {'color': cred_threshold_clr, 'width': 4}, 'thickness': 0.75, 'value': cred}}))
    fig.show()
```

Highest Confidence Prediction

In [118…
```
df = util.sort_reindex(experiments[0].loc[0, 'df'])
pred_high = df.iloc[0, :]
df.head(1)
```

Out[118]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
0	['1']	0.999917	1.0	0.1	[(8.332638946754438e-05, '0'), (1.0, '1')]	True	3.673303	3	0	-0.48654	-0.586219	0	0	1

In [119…
```
plot_prediction_uncertainty(pred_high)
```

In [137…

```
# display plotly graph in print/PDF
from IPython.display import Image
Image(filename='C:/Users/Bob/CHPC/conformal_prediction/vigilant-computing-machine/source/Notebooks/conformal_pred_high_conf.PNG')
```

Out[137]:

Lowest Confidence Prediction

In [120…
```
pred_low = df.iloc[-1, :]
df.tail(1)
```

Out[120]:

	classes	confidence	credibility	eps	p	verdict	T1	N_Biop	HypPlas	AgeMen	Age1st	N_Rels	Race	class
11999	[]	0.92209	0.07791	0.1	[(0.07791017415215398, '0'), (0.07791017415215...	False	-0.293725	2	0	2.407616	-0.416625	0	0	0

In [121…
```
plot_prediction_uncertainty(pred_low)
```

In [136…
```
# display plotly graph in print/PDF
Image(filename='C:/Users/Bob/CHPC/conformal_prediction/vigilant-computing-machine/source/Notebooks/conformal_pred_low_conf.PNG')
```

Out[136]:

Lowest Confidence Prediction (`credibility` manually modified even lower)

In [127…
```
pred_lower = pred_low.copy()
pred_lower['credibility'] = 0.04321
pred_lower = pd.DataFrame(pred_lower)
```

In [128…
```
plot_prediction_uncertainty(pred_low)
```

In [139…
```
# display plotly graph in print/PDF
Image(filename='C:/Users/Bob/CHPC/conformal_prediction/vigilant-computing-machine/source/Notebooks/conformal_pred_low_conf_lower_cred.PNG')
```

Out[139]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: