1 PiML Outcome Analysis

- Prediction Accuracy
- Weakness Detection
- Overfitting Analysis
- Prediction Uncertainty
- Robustness and Resilience
- Bias and Fairness

In [ ]:
```
!pip install piml
```

1.1 1) Example Data and Model

In [1]:
```
from piml import Experiment
exp = Experiment()
exp.data_loader(data="SimuCredit")
```

	Mortgage	Balance	Amount Past Due	Credit Inquiry	Open Trade	Delinquency	Utilization	Gender	Race	Approved
0	196153.90	2115.19	0.00	0	0	0	0.759069	1	0	1
1	149717.49	2713.77	1460.57	1	1	1	0.402820	1	0	1
2	292626.34	2209.01	0.00	0	0	0	0.684272	1	1	1
3	264812.52	21.68	0.00	0	0	0	0.037982	0	0	0
4	236374.39	1421.49	1290.85	0	0	2	0.231110	1	1	1
...	...	...	...	...	...	...	...	...	...	...
19995	236123.54	3572.34	0.00	0	0	0	0.896326	1	1	0
19996	374572.72	3560.24	0.00	0	0	0	0.648893	1	1	0
19997	279238.55	101.75	0.00	0	0	0	0.068079	0	1	0
19998	149678.27	439.46	214.36	1	0	2	0.311219	0	0	1
19999	265153.92	909.82	0.00	0	0	0	0.300862	1	1	1

20000 rows × 10 columns

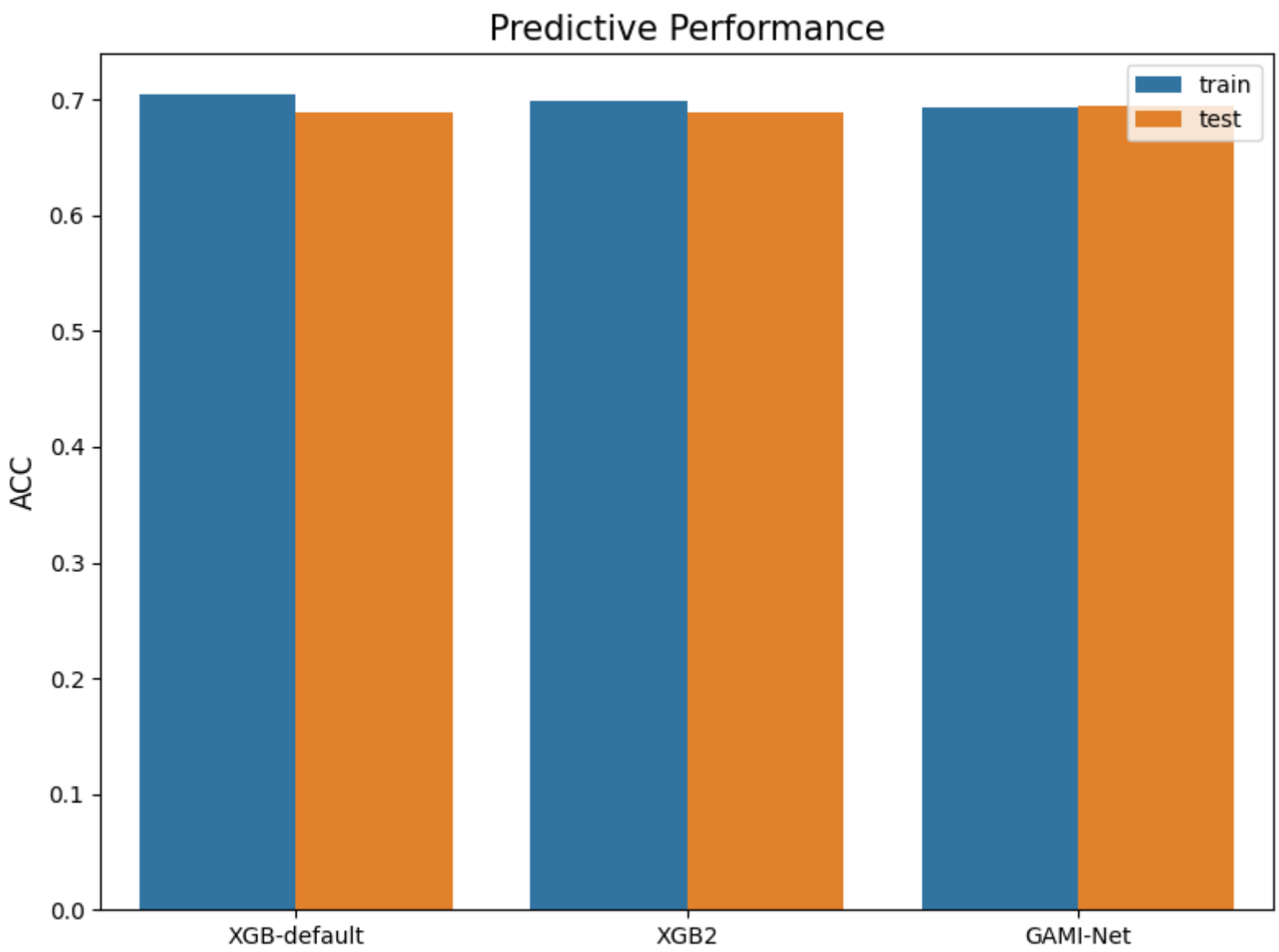
In [2]:
```
# Data Preparation
exp.data_summary(feature_exclude=["Gender", "Race"], silent=True)
exp.data_prepare(target="Approved", task_type="classification", silent=True)
```

In [3]:
```
# Build XGBoost model with max_depth 3
from xgboost import XGBClassifier
model = XGBClassifier(max_depth=3, n_estimators=1000,
                      learning_rate=0.01, random_state=0)
exp.model_train(model, name="XGB-default")
```

In [4]:
```
# Build benchmark models XGB2 and GAMI-Net

from piml.models import XGB2Classifier
from piml.models import GAMINetClassifier
exp.model_train(model=XGB2Classifier(n_estimators = 1000, eta = 0.01,
                                     mono_increasing_list=("Balance", ),
                                     mono_decreasing_list=("Utilization", )), name="XGB2")
exp.model_train(model=GAMINetClassifier(mono_increasing_list=("Balance", ),
                                     mono_decreasing_list=("Utilization", )), name="GAMI-Net")
```

In [5]:
```
exp.model_compare(models=["XGB-default", "XGB2", "GAMI-Net"], show="accuracy_plot", metric="ACC")
```



1.2 2) Prediction Accuracy

In [6]:
```
exp.model_diagnose(model="XGB-default", show='accuracy_table')
```

	ACC	AUC	F1	LogLoss	Brier
Train	0.7044	0.7746	0.7270	0.5659	0.1923
Test	0.6893	0.7560	0.7150	0.5848	0.1999
Gap	-0.0151	-0.0186	-0.0121	0.0189	0.0076

In [7]:
```
result = exp.segmented_diagnose(model="XGB-default", show="segment_table", segment_feature="Mortgage",
                        segment_method="uniform", segment_bins = 10, return_data=True)
```

	Segment ID	Feature	Segment	Size	ACC
0	0	Mortgage	[0.6005, 0.6668]	2	0.500000
1	1	Mortgage	[0.5343, 0.6005)	6	0.500000
2	2	Mortgage	[0.0042, 0.0705)	1279	0.637998
3	3	Mortgage	[0.0705, 0.1367)	1373	0.692644
4	4	Mortgage	[0.1367, 0.203)	770	0.720779
5	5	Mortgage	[0.203, 0.2693)	335	0.740299
6	6	Mortgage	[0.2693, 0.3355)	141	0.758865
7	7	Mortgage	[0.468, 0.5343)	9	0.777778
8	8	Mortgage	[0.3355, 0.4018)	56	0.785714
9	9	Mortgage	[0.4018, 0.468)	29	0.862069

In [8]:
```
result = exp.segmented_diagnose(model="XGB-default", show="segment_table", segment_feature="Mortgage",
                                segment_method="quantile", segment_bins = 10, return_data=True)
```

	Segment ID	Feature	Segment	Size	ACC
0	0	Mortgage	[0.0042, 0.0335)	400	0.632500
1	1	Mortgage	[0.0501, 0.0664)	400	0.632500
2	2	Mortgage	[0.0335, 0.0501)	400	0.647500
3	3	Mortgage	[0.0664, 0.0841)	400	0.665000
4	4	Mortgage	[0.0841, 0.1016)	400	0.677500
5	5	Mortgage	[0.1465, 0.1782)	400	0.692500
6	6	Mortgage	[0.1216, 0.1465)	400	0.715000
7	7	Mortgage	[0.1016, 0.1216)	400	0.727500
8	8	Mortgage	[0.1782, 0.2317)	400	0.730000
9	9	Mortgage	[0.2317, 0.6668]	400	0.772500

In [9]:
```
# Automatic segmentation through XGB1 surrogate modeling (|residual| ~ x1)
result = exp.segmented_diagnose(model="XGB-default", show="segment_table", segment_feature="Utilization",
                                segment_method="auto", return_data=True)
```

	Segment ID	Feature	Segment	Size	ACC
0	0	Utilization	[-inf, 0.309)	1537	0.648016
1	1	Utilization	[0.309, 0.3926)	400	0.675000
2	2	Utilization	[0.8021, 0.8878)	294	0.704082
3	3	Utilization	[0.3926, 0.4564)	234	0.713675
4	4	Utilization	[0.4564, 0.8021)	1055	0.714692
5	5	Utilization	[0.8988, inf]	438	0.755708
6	6	Utilization	[0.8878, 0.8988)	42	0.761905

In [10]:
```
# Joint segmentation (shown top 10 worst performing segments) (|residual| ~ x1 + x2 + ... )
result = exp.segmented_diagnose(model="XGB-default", show="segment_table",
                                segment_method="auto", metric="ACC",
                                return_data=True)
```

	Segment ID	Feature	Segment	Size	ACC
0	0	Balance	[0.1842, 0.2088)	61	0.540984
1	1	Balance	[0.0063, 0.011)	311	0.578778
2	2	Mortgage	[0.0455, 0.064)	472	0.610169
3	3	Mortgage	[0.0256, 0.0391)	312	0.637821
4	4	Mortgage	[0.2859, 0.316)	56	0.642857
5	5	Utilization	[-inf, 0.309)	1537	0.648016
6	6	Mortgage	[-inf, 0.0256)	220	0.650000
7	7	Balance	[0.011, 0.0184)	446	0.650224
8	8	Amount Past Due	[-inf, 0.0006)	2343	0.654716
9	9	Mortgage	[0.0391, 0.0455)	144	0.659722

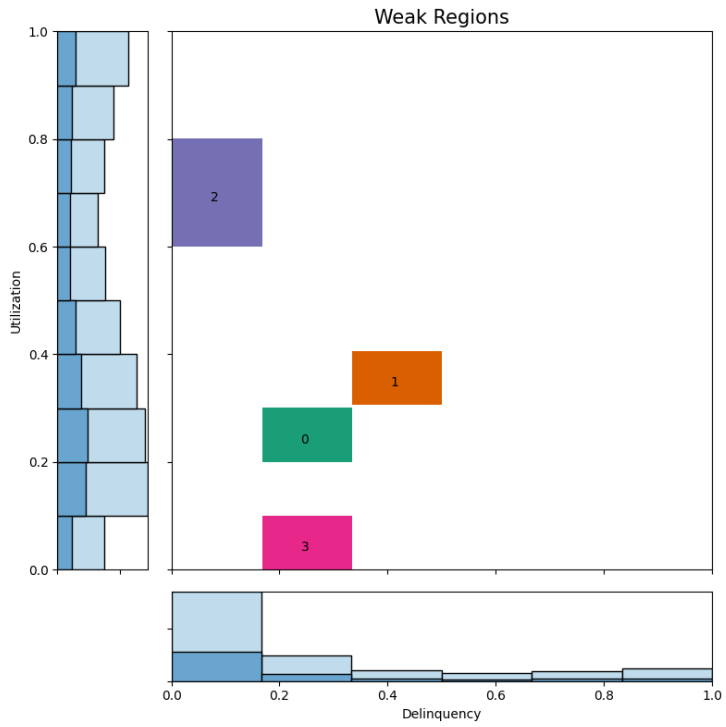
1.3 3) Weakness Detection

In [11]:
```
# Choose XGB-default: WeakSpot, 1D (Delinquency) with ACC/Threshold 1.0; 2D (Deliquency, Utilization) with ACC/
exp.model_diagnose()
```

XGB-default

Accuracy | WeakSpot | Overfit | Reliability | Robustness | Resilience

Feature 1: Delinquency
Feature 2: Utilization
Min Sample: 20
Method: Histogram slicir
Threshold: 1.1
Dataset: train
Metric: ACC



	[Delinquency	Delinquency)	[Utilization	Utilization)	#Tes
0	0.1667	0.3333	0.2000	0.3000	9
1	0.3333	0.5000	0.3060	0.4051	3
2	0.0000	0.1667	0.6006	0.8003	33
3	0.1667	0.3333	0.0000	0.1000	4

Shown in original scale: ☐

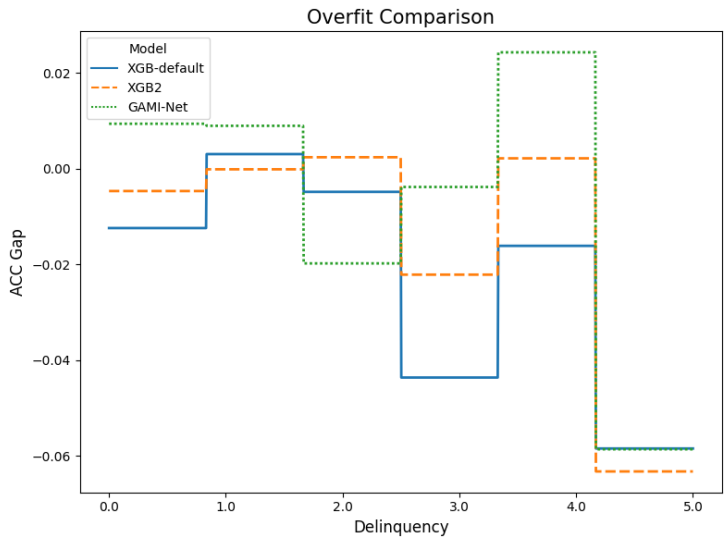
1.4 4) Overfitting Analysis

In [14]:
```
# Choose XGB-default, XGB2, GAMI-Net: Overfit (Delinquency, AUC)
exp.model_compare()
```

XGB-default | XGB2 | GAMI-Net

Accuracy | Overfit | Reliability | Robustness | Resilience

Feature: Delinquency Metric: ACC



Shown in original scale: ☑

1.5 5) Prediction Uncertainty

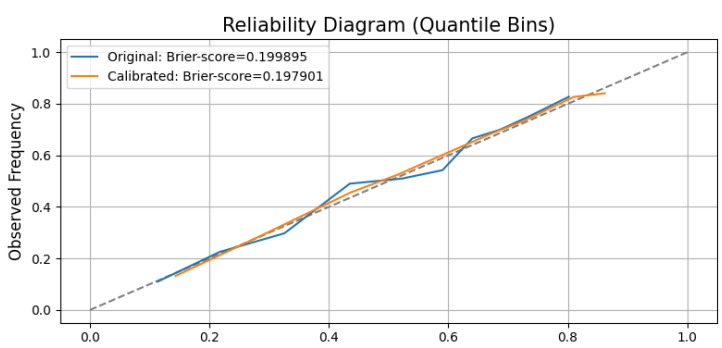
In [15]:
```
# Choose XGB-default, Reliablity: Bandwidth Threshold = 1
exp.model_diagnose()
```

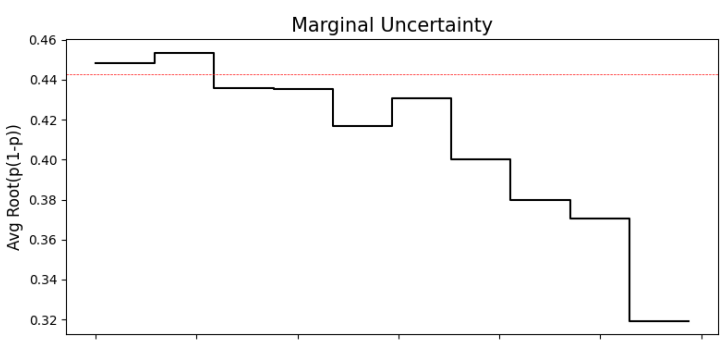
XGB-default

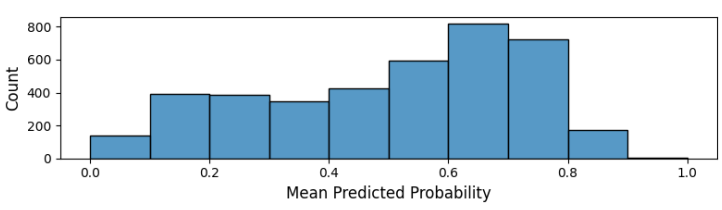
Accuracy | WeakSpot | Overfit | **Reliability** | Robustness | Resilience

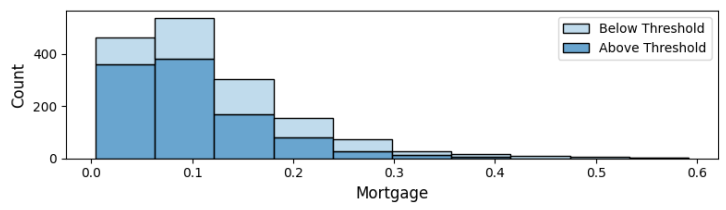
Expected Coverage: 0.9 Feature: Mortgage Bins: 10

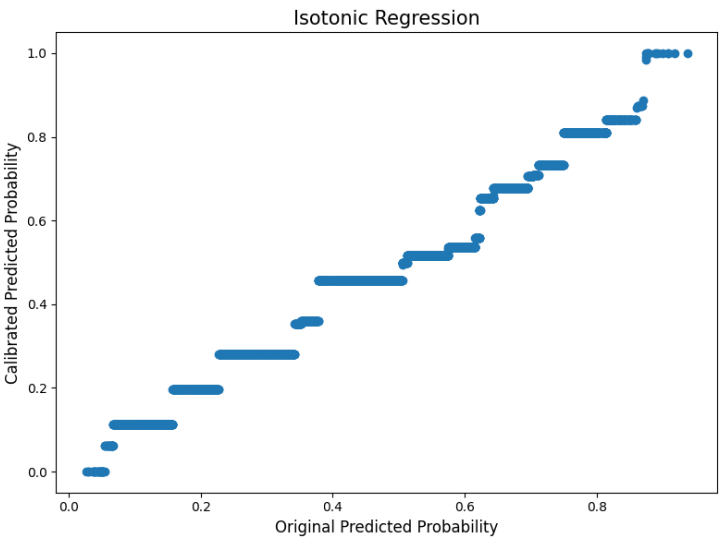
Bandwidth Threshold: 1 Distance Metric: PSI

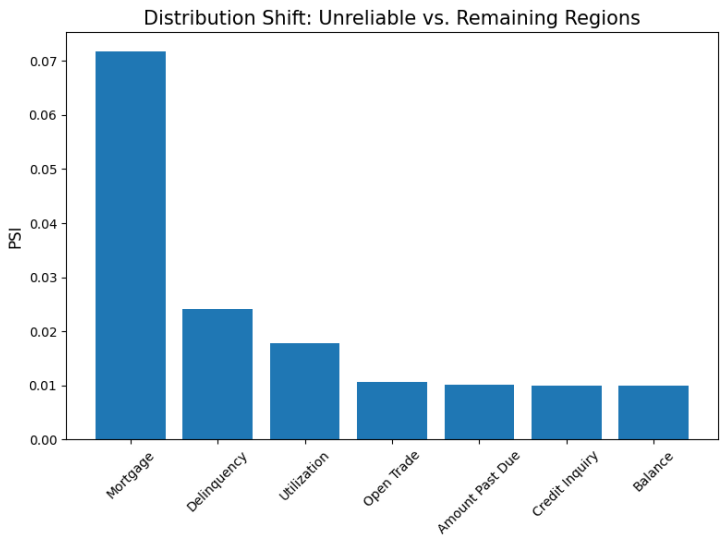












Shown in original scale: ☐

1.6 6) Robustness Test

In [16]:
```
# Choose XGB-default, XGB2, GAMI-Net: Robustness: Noise step 0.05
exp.model_compare()
```

XGB-default | XGB2 | GAMI-Net

Accuracy | Overfit | Reliability | **Robustness** | Resilience

Perturb: All Features | Noise Scale: Raw Scale | Noise Step: 0.05

Metric: AUC | Worst Ratio: 0.4

1.7 7) Reslience Test

In [17]:
```
# Choose XGB-default, Resilience: worst-sample
exp.model_diagnose()
```

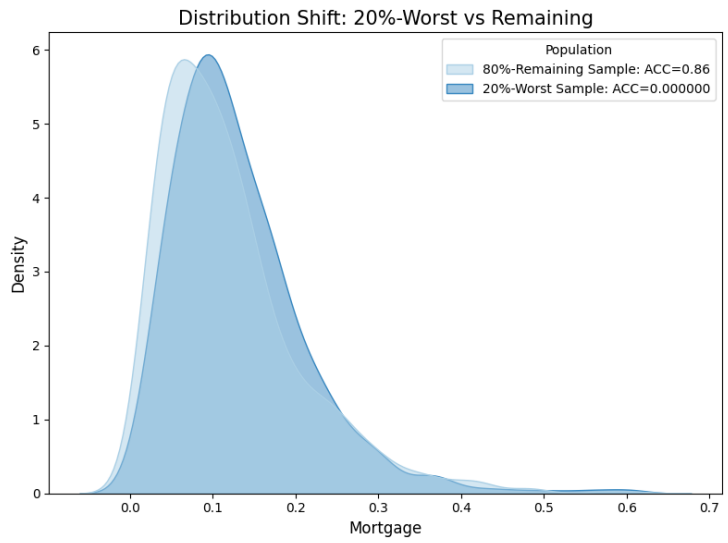
XGB-default

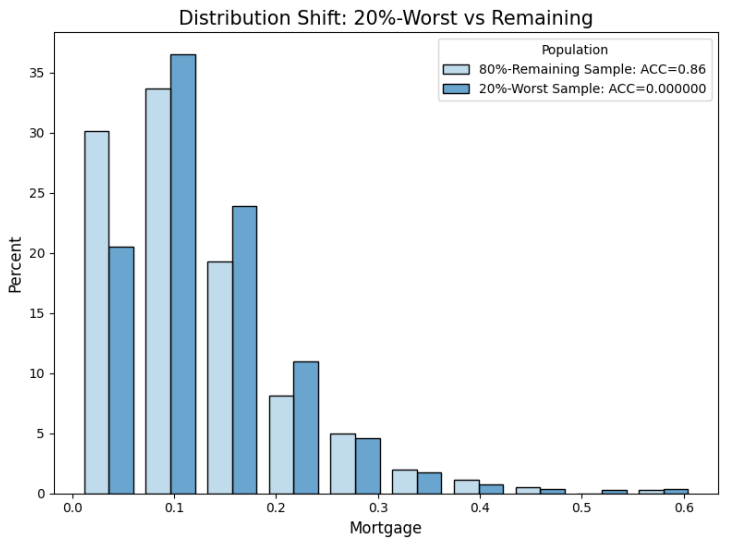
Accuracy | WeakSpot | Overfit | Reliability | Robustness | Resilience

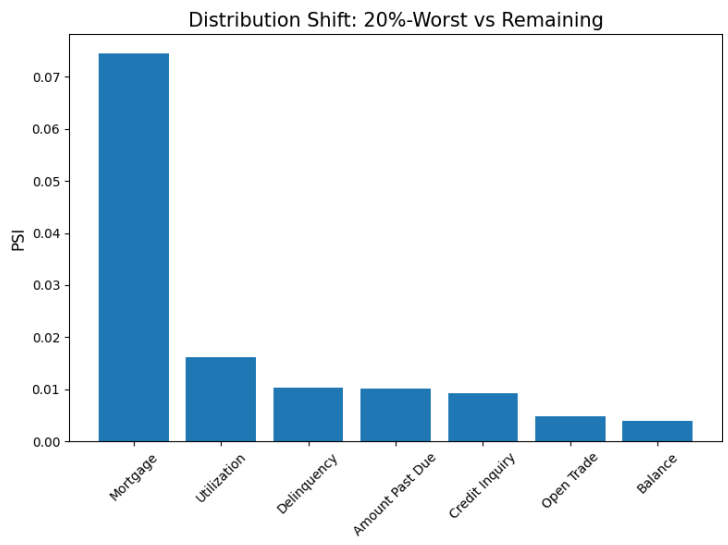
Method: worst-sample | Immutable Feature: None | Worst Ratio: 0.2

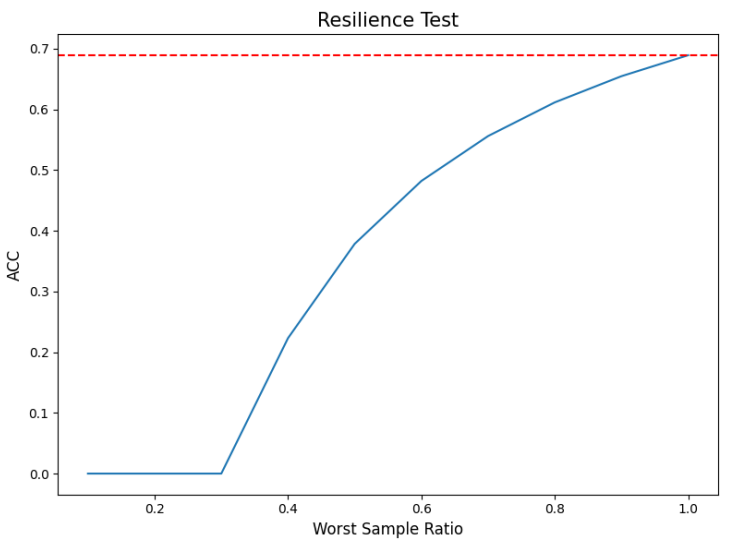
Metric: ACC | Plot Feature: Mortgage | PSI Buckets: Uniform

Distance Metric: PSI









Shown in original scale:

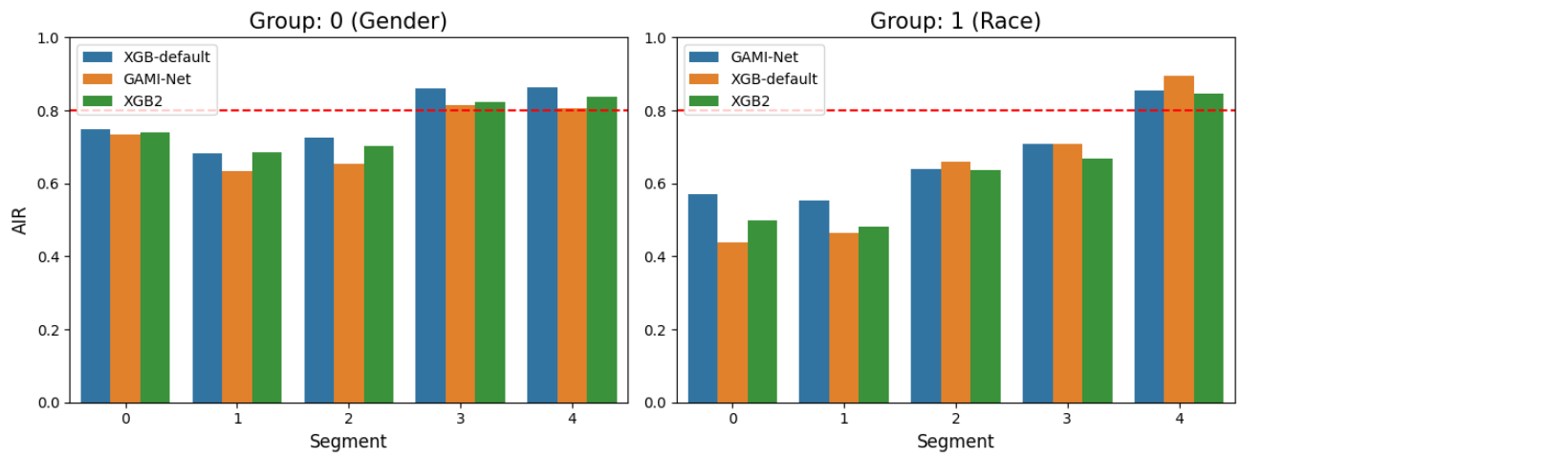
1.8 8) Bias and Fairness

In [18]:
```
# Choose XGB-default, XGB2, GAMI-Net: Group Setting Gender/Race Reference 1.0
exp.model_fairness_compare()
```

XGB-default | XGB2 | GAMI-Net

Setting | Metrics | Segment

Metrics: AIR | Threshold: 0.8 | Feature: Balance | Bins: 5



Segment	Lower Bound	Upper Bound	Group Index	XGB-default_AIR	XGB2_AIR	GAMI-Net_AIR
0	0.970000	306.610000	0	0.749742	0.740113	0.735404
0	0.970000	306.610000	1	0.438113	0.497803	0.569818
1	306.620000	601.400000	0	0.683004	0.686304	0.634271
1	306.620000	601.400000	1	0.465392	0.482222	0.553344
2	601.470000	1027.230000	0	0.724709	0.702708	0.652656
2	601.470000	1027.230000	1	0.659653	0.636542	0.638701
3	1027.290000	1864.900000	0	0.860570	0.823053	0.813099
3	1027.290000	1864.900000	1	0.708812	0.668660	0.707100
4	1864.940000	20384.870000	0	0.863693	0.836753	0.804388
4	1864.940000	20384.870000	1	0.895253	0.845336	0.854540

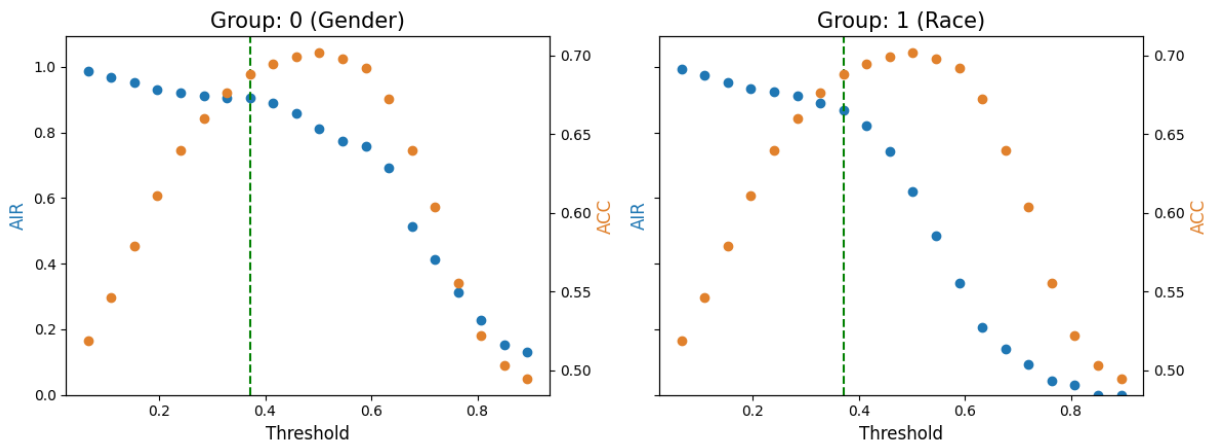
In [19]:
```
# Choose XGB-default: Group Setting Gender/Race Reference 1.0
exp.model_fairness()
```

XGB-default

Setting | Metrics | Segmented | Binning | Thresholding

Fairness Metric: AIR | Performance Metric: ACC | Grid Size: 20

Threshold: 0.37



Group Index	Group Category	Reference Group	Protected Group	Threshold	AIR	ACC
0	Gender	1.0	0.0	0.37	0.904196	0.688250
1	Race	1.0	0.0	0.37	0.867701	0.688250