本项目旨在探索不同类别关键审计事项下，企业基本面特征对审计判断的差异化影响。具体聚焦于五类高频KAM：收入确认（Revenue）、其他事项（Other）、应收账款（Accounts Receivable）、无形资产（Intangible Assets）和存货（Inventories）。

基于上市公司公开披露的审计报告与财务数据，构建了分类标签，并选取一系列公司层面的解释变量，包括规模（SIZE）、上市年限（AGE）、账面市值比（BM）、销售增长率（SALES-GROWTH）、杠杆率（LEV）、盈利能力（ROA）、亏损状态（LOSS）等。

在此基础上，采用 XGBoost 模型对每一类KAM分别建模，利用其特征重要性（Feature Importance）和SHAP值分析，识别在不同审计事项下哪些变量具有更强的预测能力，进而比较“什么因素让审计师更关注收入？什么因素更易触发无形资产减值审计？”等问题上的差异。

代码准备

In [63]:
```
from IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast_node_interactivity = "all"

import matplotlib_inline

matplotlib_inline.backend_inline.set_matplotlib_formats("svg")
```

In [64]:
```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_multilabel_classification
from sklearn.model_selection import RandomizedSearchCV

# randint:离散均匀分布
# uniform: 连续均匀分布
from scipy.stats import uniform
from scipy.stats import randint

from sklearn.metrics import f1_score
import pandas as pd
import shap
from PyALE import ale
```

导入数据

df_B是处理缺失值后的控制变量表与y合并的表

In [65]:
```
df_B=pd.read_excel("D:\桌面\data\df_B.xlsx",header=0)
df_B.head()
```

```
invalid escape sequence '\d'
invalid escape sequence '\d'
invalid escape sequence '\d'
```

Out[65]:

	证券代码	证券简称	统计截止日期	报表类型	SIZE	AGE	BM	SALES-GROWTH	SEGMENTS	LEV	...
0	2	万科A	2016	A	27.445504	25.0	0.182068	1.920584	4.276666	0.805367	...
1	2	万科A	2017	A	27.784040	26.0	0.143285	1.660109	4.174387	0.839813	...
2	2	万科A	2018	A	28.055360	27.0	0.151756	0.736779	4.304065	0.845856	...
3	2	万科A	2019	A	28.179102	28.0	0.149493	0.701988	4.317488	0.843590	...
4	2	万科A	2020	A	28.256519	29.0	0.188550	0.866897	4.369448	0.812835	...

5 rows × 25 columns

In [66]:

```
df_B.isnull().sum()
df_B.dtypes
```

Out[66]:

```
证券代码                0
证券简称                6
统计截止日期              0
报表类型                6
SIZE                    6
AGE                     6
BM                      6
SALES-GROWTH            6
SEGMENTS                6
LEV                     6
ROA                     6
LOSS                    6
ZSCORE                  0
DEF-REVENUES            6
RECEIVABLES             6
INVENTORY               6
PPE                     6
INTANGIBLE              6
IMPAIR                  6
LIT-RISK                6
Revenue              4625
Accounts-receivable  4625
intangible-assets    4625
other                4625
inventories          4625
dtype: int64
```

Out[66]:
```
证券代码                 int64
证券简称                object
统计截止日期               int64
报表类型                object
SIZE                   float64
AGE                    float64
BM                     float64
SALES-GROWTH           float64
SEGMENTS               float64
LEV                    float64
ROA                    float64
LOSS                   float64
ZSCORE                  object
DEF-REVENUES           float64
RECEIVABLES            float64
INVENTORY              float64
PPE                    float64
INTANGIBLE             float64
IMPAIR                 float64
LIT-RISK               float64
Revenue                float64
Accounts-receivable    float64
intangible-assets      float64
other                  float64
inventories            float64
dtype: object
```

处理缺失

处理缺失值后的数据表

删除缺失所有y值的样本；x下,缺失值较少，填充为0

In [67]:
```
df=df_B.dropna(subset=['Revenue','Accounts-receivable','intangible-assets','othe

df= df.fillna(0)
```

In [68]:
```
df.isnull().sum()
```

Out[68]:
```
证券代码                   0
证券简称                   0
统计截止日期                 0
报表类型                   0
SIZE                   0
AGE                    0
BM                     0
SALES-GROWTH           0
SEGMENTS               0
LEV                    0
ROA                    0
LOSS                   0
ZSCORE                 0
DEF-REVENUES           0
RECEIVABLES            0
INVENTORY              0
PPE                    0
INTANGIBLE             0
IMPAIR                 0
LIT-RISK               0
Revenue                0
Accounts-receivable    0
intangible-assets      0
other                  0
inventories            0
dtype: int64
```

数据表相关信息

In [69]:
```
df.head()
df.columns

df.shape
df.dtypes
```

Out[69]:

	证券代码	证券简称	统计截止日期	报表类型	SIZE	AGE	BM	SALES-GROWTH	SEGMENTS	LEV	...
0	2	万科A	2016	A	27.445504	25.0	0.182068	1.920584	4.276666	0.805367	...
1	2	万科A	2017	A	27.784040	26.0	0.143285	1.660109	4.174387	0.839813	...
2	2	万科A	2018	A	28.055360	27.0	0.151756	0.736779	4.304065	0.845856	...
3	2	万科A	2019	A	28.179102	28.0	0.149493	0.701988	4.317488	0.843590	...
4	2	万科A	2020	A	28.256519	29.0	0.188550	0.866897	4.369448	0.812835	...

5 rows × 25 columns

Out[69]:

```
Index(['证券代码', '证券简称', '统计截止日期', '报表类型', 'SIZE', 'AGE', 'BM',
'SALES-GROWTH',
       'SEGMENTS', 'LEV', 'ROA', 'LOSS', 'ZSCORE', 'DEF-REVENUES',
       'RECEIVABLES', 'INVENTORY', 'PPE', 'INTANGIBLE', 'IMPAIR', 'LIT-RISK',
       'Revenue', 'Accounts-receivable', 'intangible-assets', 'other',
       'inventories'],
      dtype='object')
```

Out[69]:

```
(23748, 25)
```

Out[69]:
```
证券代码                   int64
证券简称                  object
统计截止日期                  int64
报表类型                  object
SIZE                   float64
AGE                    float64
BM                     float64
SALES-GROWTH           float64
SEGMENTS               float64
LEV                    float64
ROA                    float64
LOSS                   float64
ZSCORE                  object
DEF-REVENUES           float64
RECEIVABLES            float64
INVENTORY              float64
PPE                    float64
INTANGIBLE             float64
IMPAIR                 float64
LIT-RISK               float64
Revenue                float64
Accounts-receivable    float64
intangible-assets      float64
other                  float64
inventories            float64
dtype: object
```

转换x类型

In [70]:
```
df['ZSCORE'] = pd.to_numeric(df['ZSCORE'], errors='coerce')

df['ZSCORE'].fillna(0, inplace=True)
```

数据生成

In [71]:
```
X=df.drop(['证券代码', '证券简称', '统计截止日期', '报表类型','Revenue', 'Account
y=df[['Revenue', 'Accounts-receivable', 'intangible-assets','other', 'inventorie
```

转换y数据类型为整数型

In [72]:
```
y.dtypes
```

Out[72]:
```
Revenue                float64
Accounts-receivable    float64
intangible-assets      float64
other                  float64
inventories            float64
dtype: object
```

In [73]:
```
y=y.astype('int8')
```

调整y的顺序

In [74]:
```
column_sums = y.sum(axis=0)
sorted_columns = column_sums.sort_values(ascending=False).index
y = y[sorted_columns]
y.head()
```

Out[74]:

	Revenue	other	Accounts-receivable	intangible-assets	inventories
0	1	1	0	0	1
1	1	1	0	0	1
2	1	1	0	0	1
3	1	1	0	0	1
4	1	1	0	0	1

分割训练与测试集

In [75]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

模型调参

In [76]:
```
params = {"max_depth": [2,3,4,5,6,7],"learning_rate": uniform(0.0001, 0.1),"n_es
rscv_clf = xgb.XGBClassifier(tree_method="hist", multi_strategy="one_output_per_
grid = RandomizedSearchCV(
    estimator=rscv_clf,
    param_distributions=params,
    n_iter=10,
    scoring="f1_macro",
    n_jobs=-1,
    cv=5,
    random_state=0
)
grid.fit(X_train, y_train)
```

Out[76]:
```
▸ RandomizedSearchCV
▸ estimator: XGBClassifier
    ▸ XGBClassifier
```

调参结果

In [77]:
```
print("Best parameters: {}".format(grid.best_params_))
print("Best cross-validation score: {:.3f}".format(grid.best_score_))
```

```
Best parameters: {'gamma': 0, 'learning_rate': 0.08019107519796444, 'max_depth':
7, 'min_child_weight': 5, 'n_estimators': 247}
Best cross-validation score: 0.578
```

重新训练模型

In [78]:
```python
def scoring_clf(y_true, y_pred):
    print("-" * 10, "\n")
    print("f1_micro:", f1_score(y_true, y_pred,average="micro",zero_division=0))
    print("f1_macro:", f1_score(y_true, y_pred,average="macro",zero_division=0))
    print("f1_weighted:", f1_score(y_true, y_pred,average="weighted",zero_divisi
    print("f1_samples:", f1_score(y_true, y_pred,average="samples",zero_division
```

拟合数据、预测数据、评估模型

In [79]:
```python
gbclf = xgb.XGBClassifier(
    tree_method="hist", multi_strategy="one_output_per_tree",
    learning_rate=grid.best_params_["learning_rate"],
    max_depth=grid.best_params_["max_depth"],
    n_estimators=grid.best_params_["n_estimators"],
    gamma=grid.best_params_['gamma'],
    min_child_weight=grid.best_params_['min_child_weight'],

    random_state=0
)


gbclf.fit(X_train, y_train)


y_train_pred = gbclf.predict(X_train)
y_test_pred = gbclf.predict(X_test)


scoring_clf(y_train, y_train_pred)
scoring_clf(y_test, y_test_pred)
```

Out[79]:

▾ XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=Non
e,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.080191075197964
44,
```

```
----------

f1_micro: 0.8613666503804007
f1_macro: 0.8056215974797315
f1_weighted: 0.8499652793998164
f1_samples: 0.8533402847751725
----------

f1_micro: 0.7120191758931628
f1_macro: 0.5833931921082863
f1_weighted: 0.678736097936342
f1_samples: 0.7028887218045112
```

模型解释

特征重要性

In [80]:
```
featureimportance=pd.DataFrame(
    {"gain": gbclf.get_booster().get_score(importance_type='gain'),
     "weight": gbclf.get_booster().get_score(importance_type='weight'),
     "cover": gbclf.get_booster().get_score(importance_type='cover')}
)
featureimportance
```

Out[80]:

	gain	weight	cover
SIZE	5.229168	5948.0	480.742615
AGE	6.713708	3700.0	450.662231
BM	4.150448	4791.0	357.987213
SALES-GROWTH	3.838553	4624.0	396.202026
SEGMENTS	5.151576	4972.0	384.488220
LEV	5.226228	5452.0	425.654205
ROA	4.659832	5228.0	464.028595
RECEIVABLES	10.350252	5828.0	459.856689
INVENTORY	4.226520	5402.0	364.264923
PPE	4.979679	6237.0	434.678162
INTANGIBLE	4.814849	6247.0	378.556183
IMPAIR	45.066063	483.0	898.429321
LIT-RISK	5.960119	422.0	606.272095

全局解释

In [81]:
```
shap_values = shap.TreeExplainer(gbclf).shap_values(X)
```

```
[12:17:34] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\c_api\c_api.cc:1240: Saving into deprecated binary model format, please consider using `json` or `ubj`. Model format will default to JSON in XGBoost 2.2 if not specified.
```

各个y下，X的影响力

In [82]:

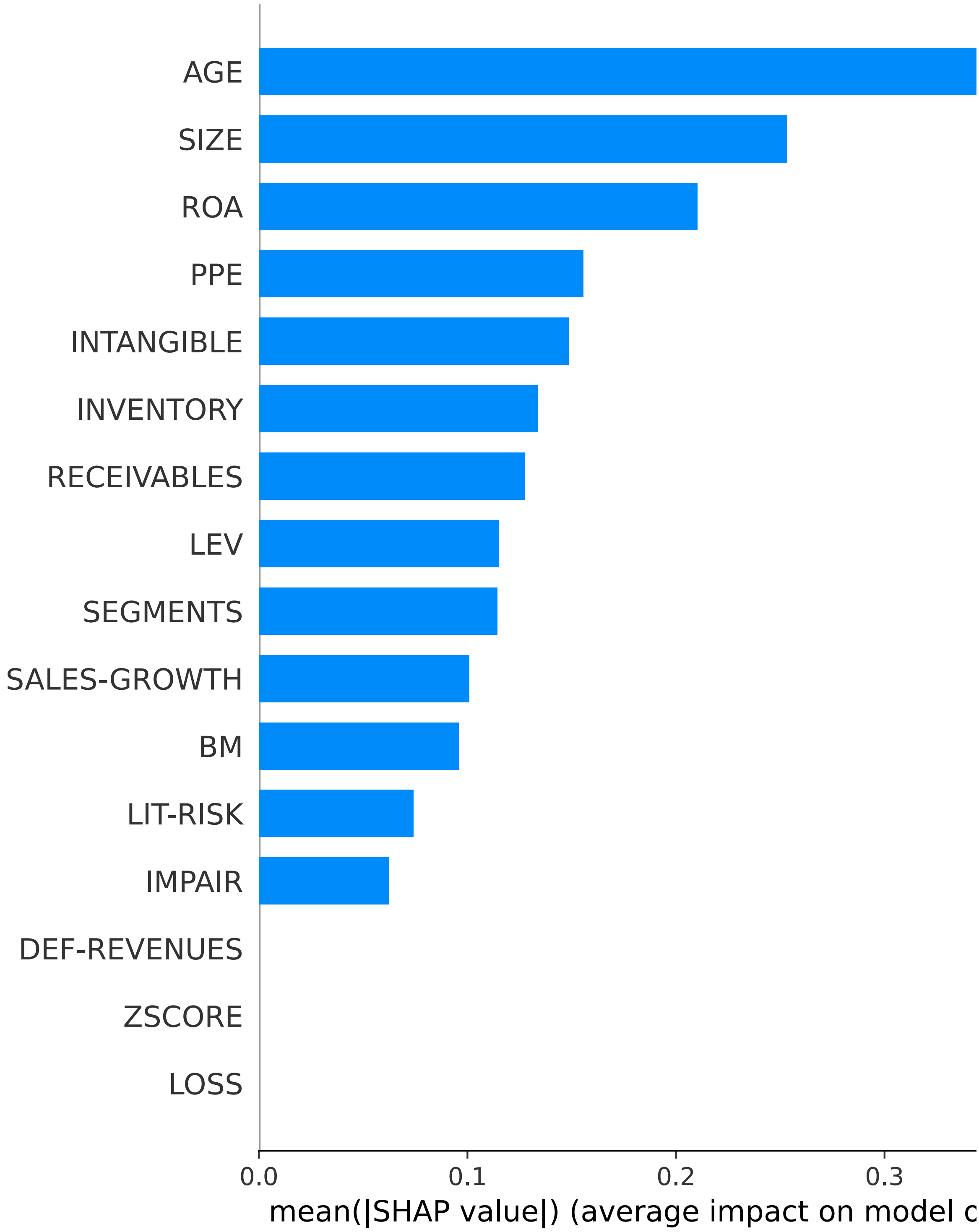
```
graph_names = ['Revenue', 'other','Accounts-receivable', 'intangible-assets', 'i

for i, j in enumerate(graph_names):
    print(f"i={i}, j={j}")
```

```
i=0, j=Revenue
i=1, j=other
i=2, j=Accounts-receivable
i=3, j=intangible-assets
i=4, j=inventories
```

y0

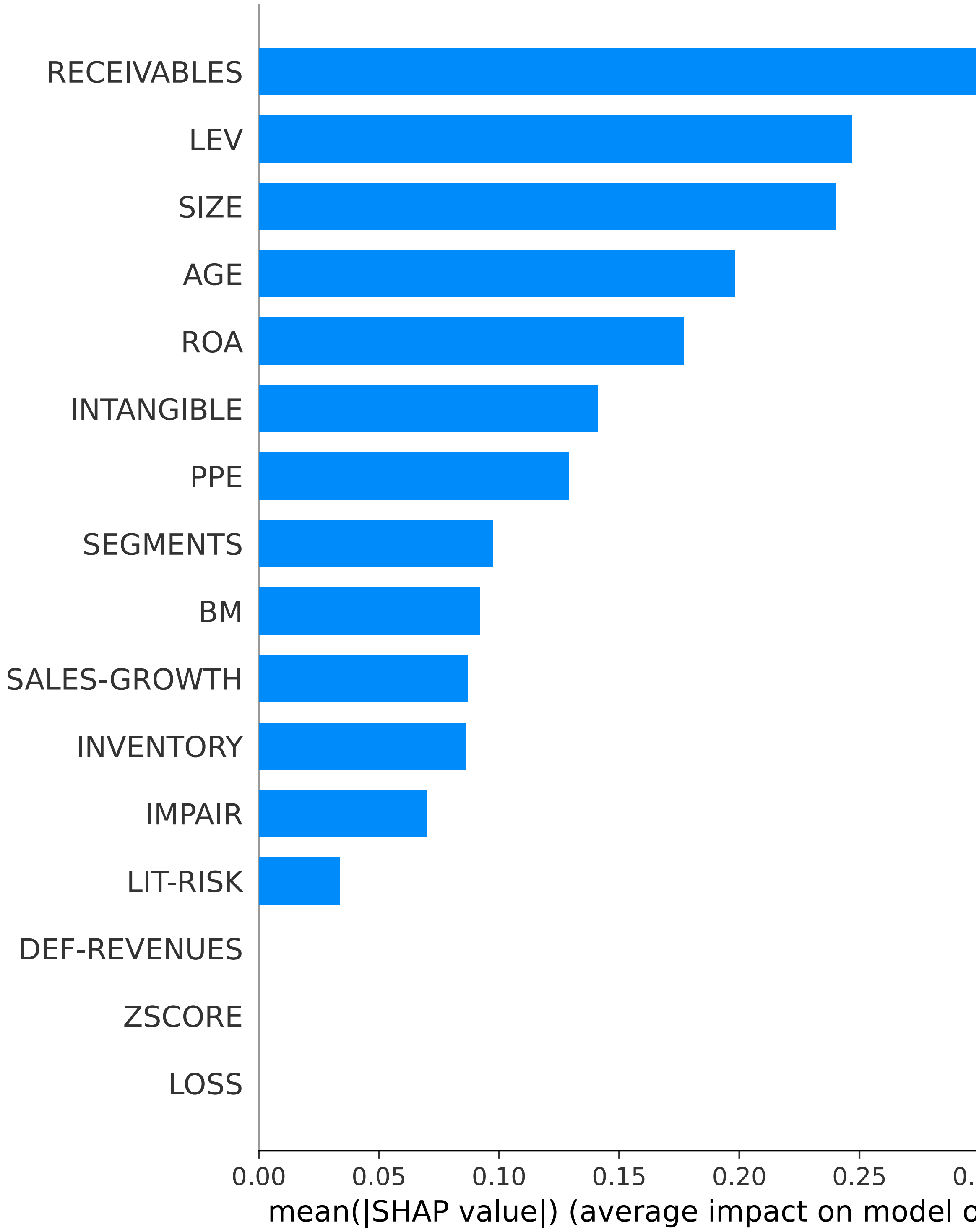
In [83]:

```
shap.summary_plot(shap_values[0], X, plot_type="bar")
```



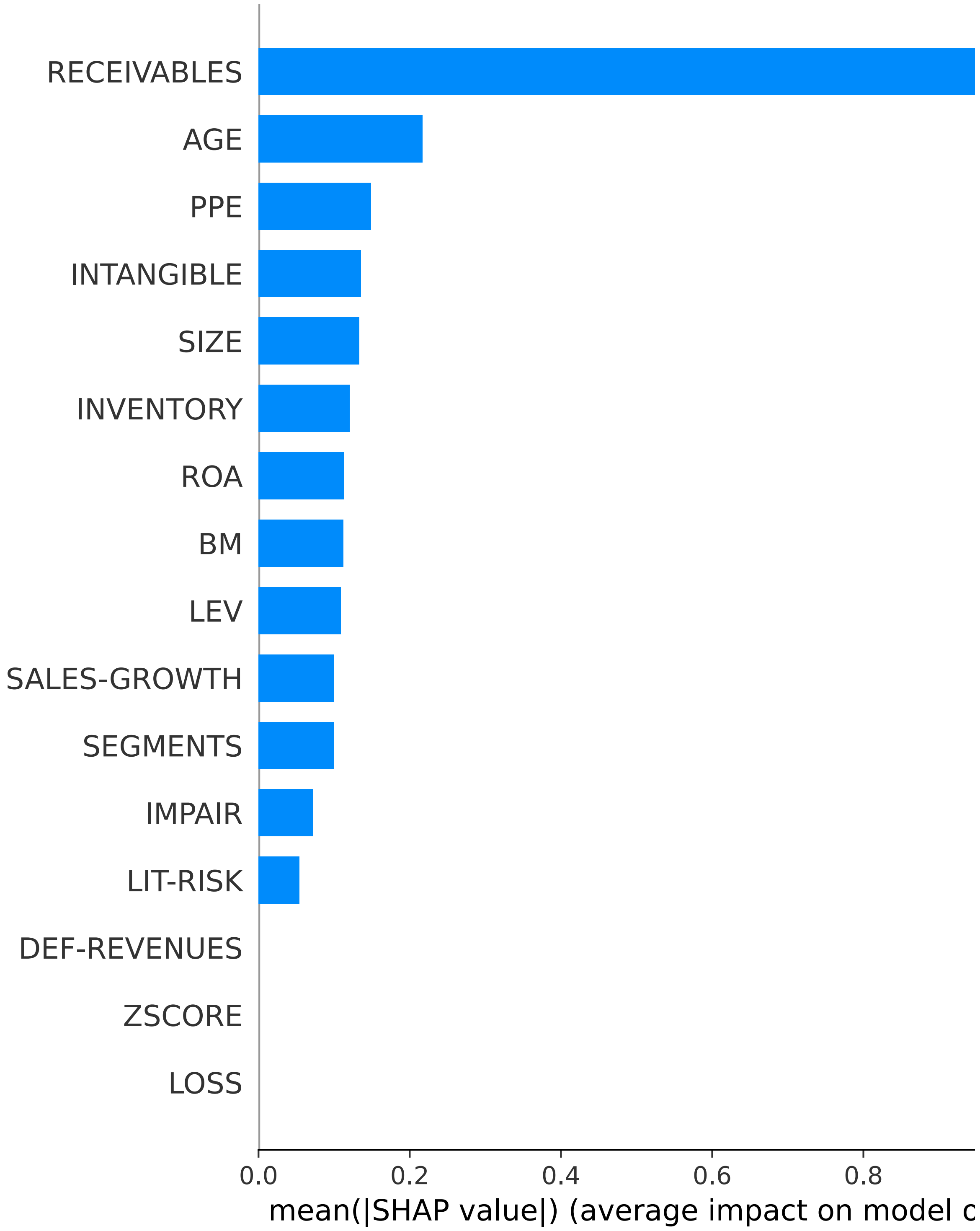
y1

In [84]:
```
shap.summary_plot(shap_values[1], X, plot_type="bar")
```



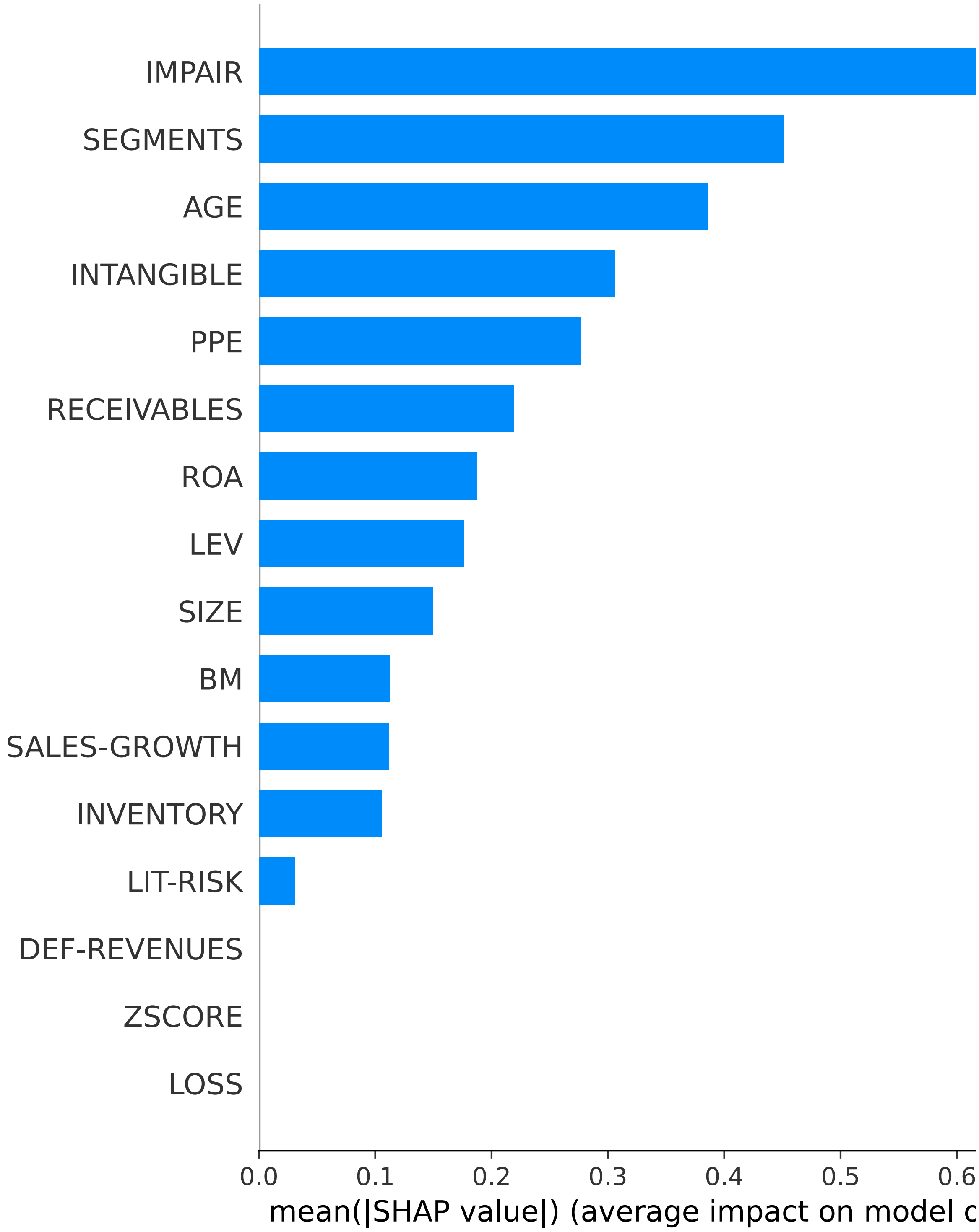
y2

In [85]:
```
shap.summary_plot(shap_values[2], X, plot_type="bar")
```



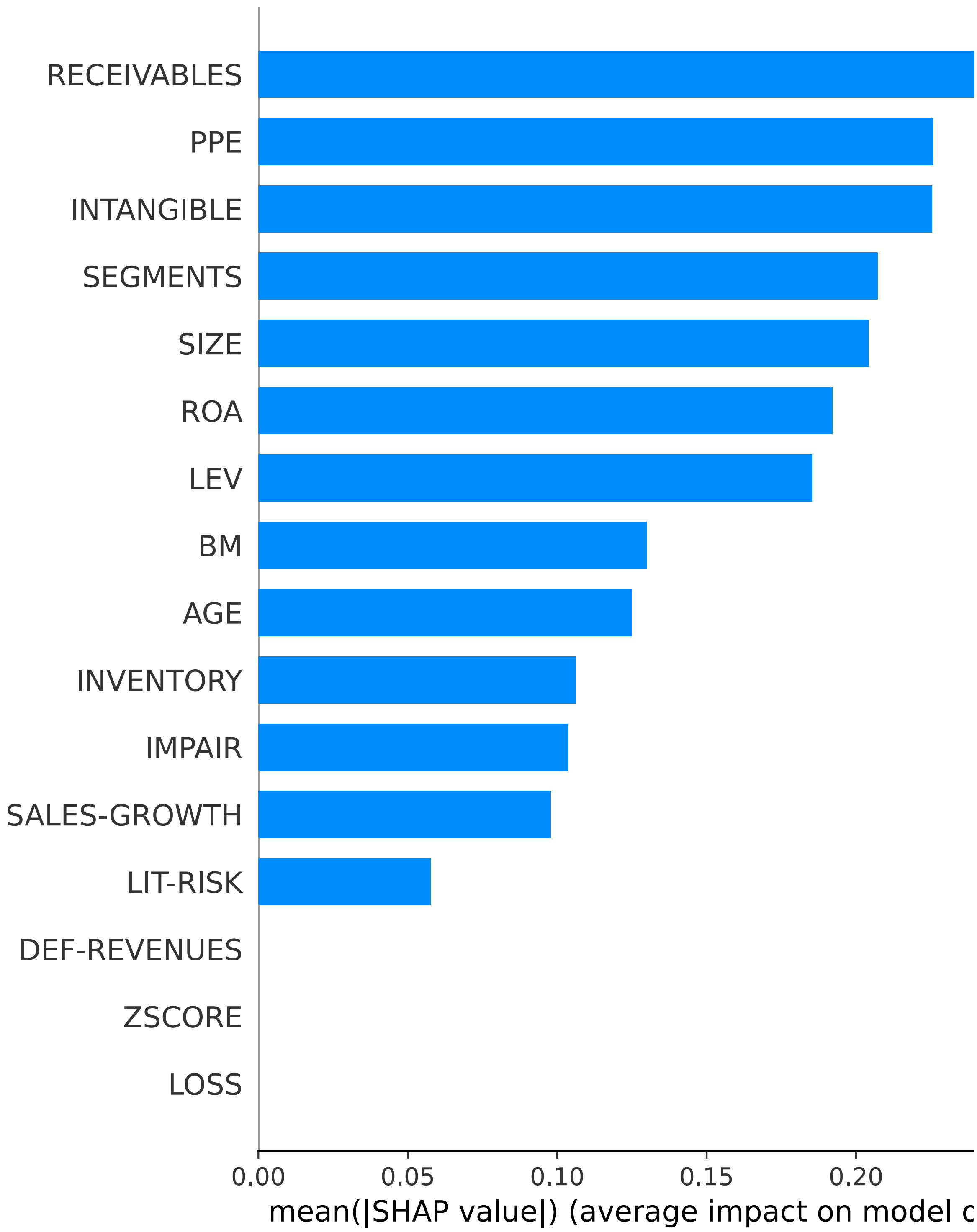
y3

In [86]:
```
shap.summary_plot(shap_values[3], X, plot_type="bar")
```



y4

In [87]:
```
shap.summary_plot(shap_values[4], X, plot_type="bar")
```



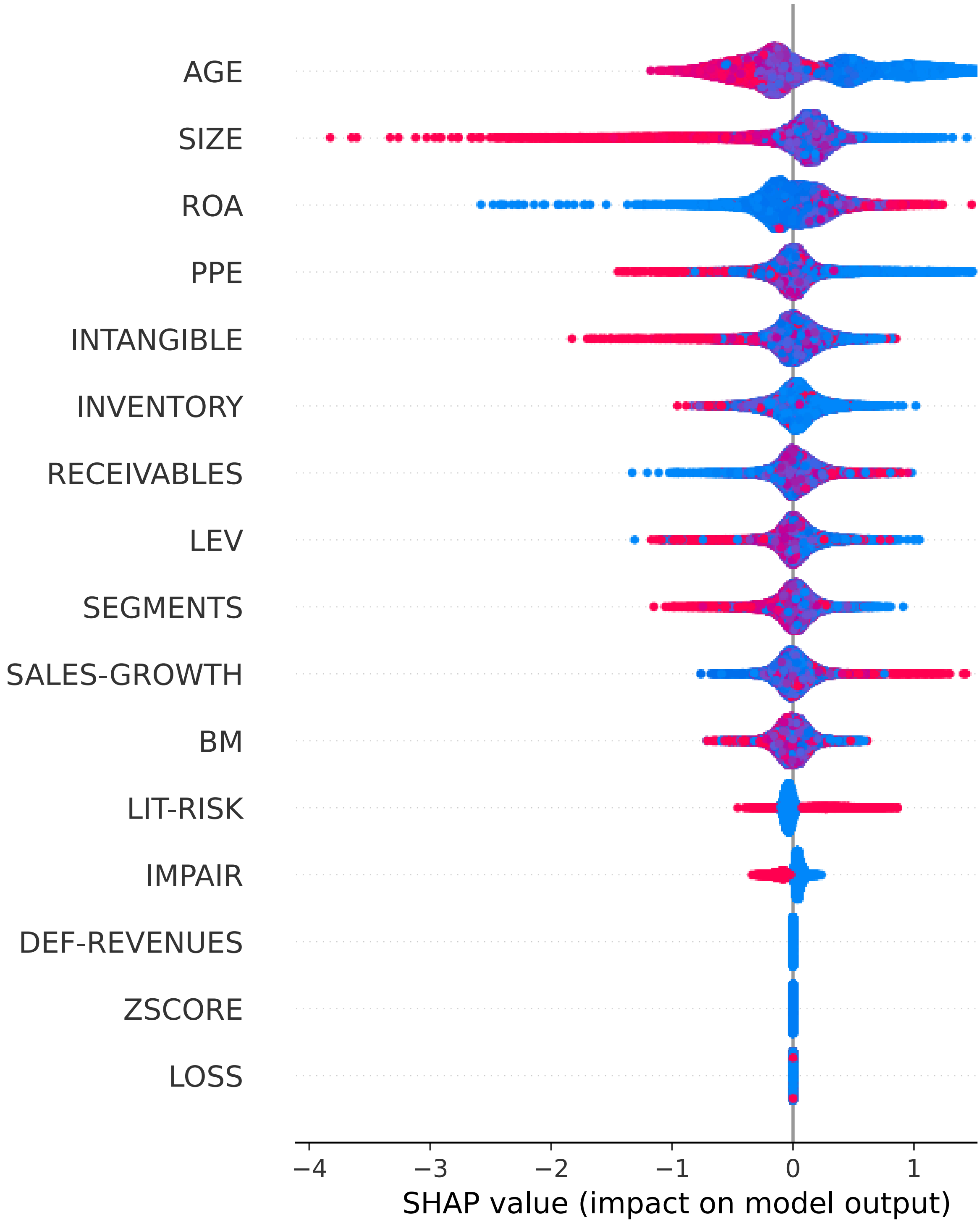
特征与目标变量的关系

概要图

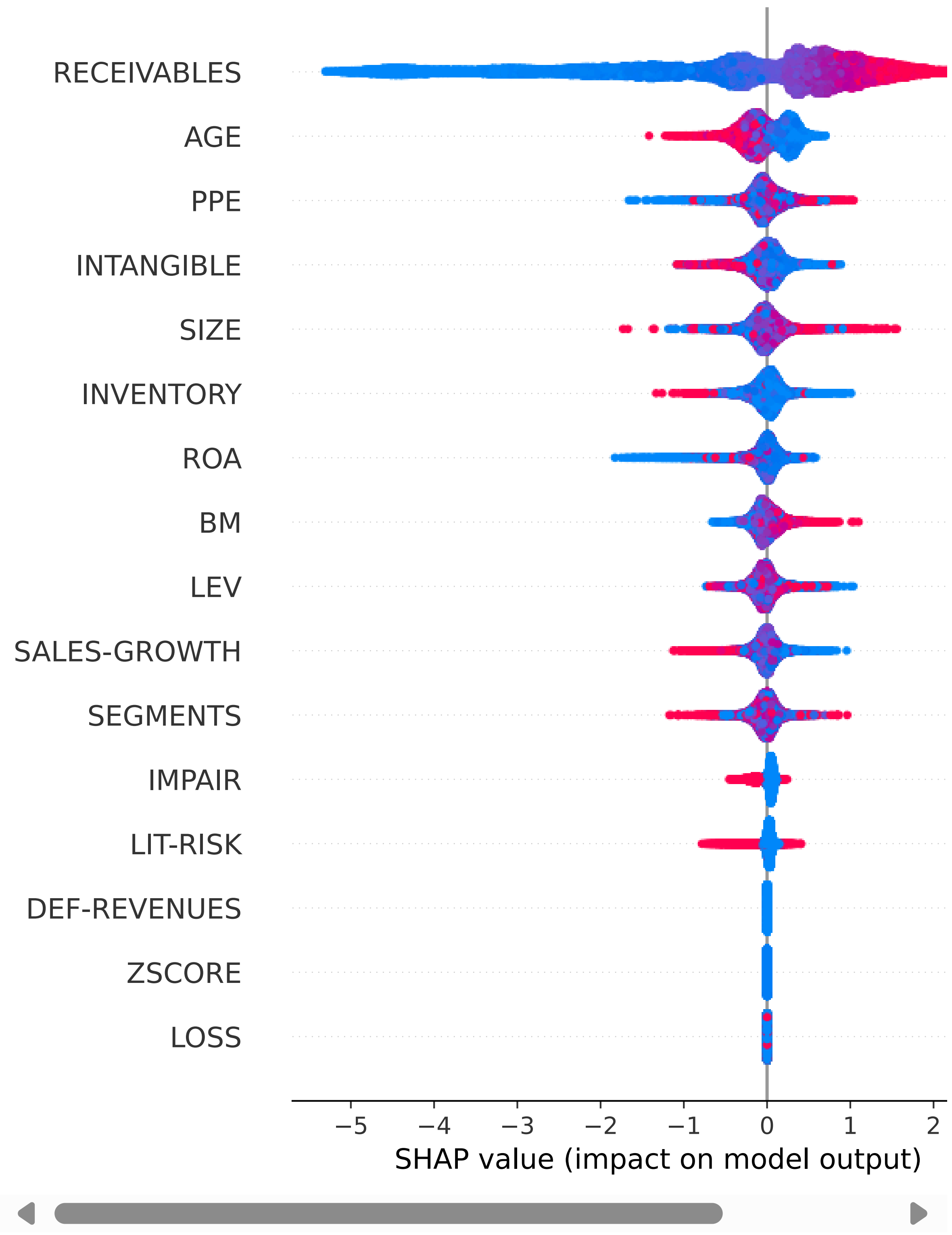
In [88]:
```
graph_names = ['Revenue', 'other','Accounts-receivable', 'intangible-assets', 'i

for i, j in enumerate(graph_names):
    print(f"i={i}, j={j}")

shap.summary_plot(shap_values[0], X)
shap.summary_plot(shap_values[1], X)
```

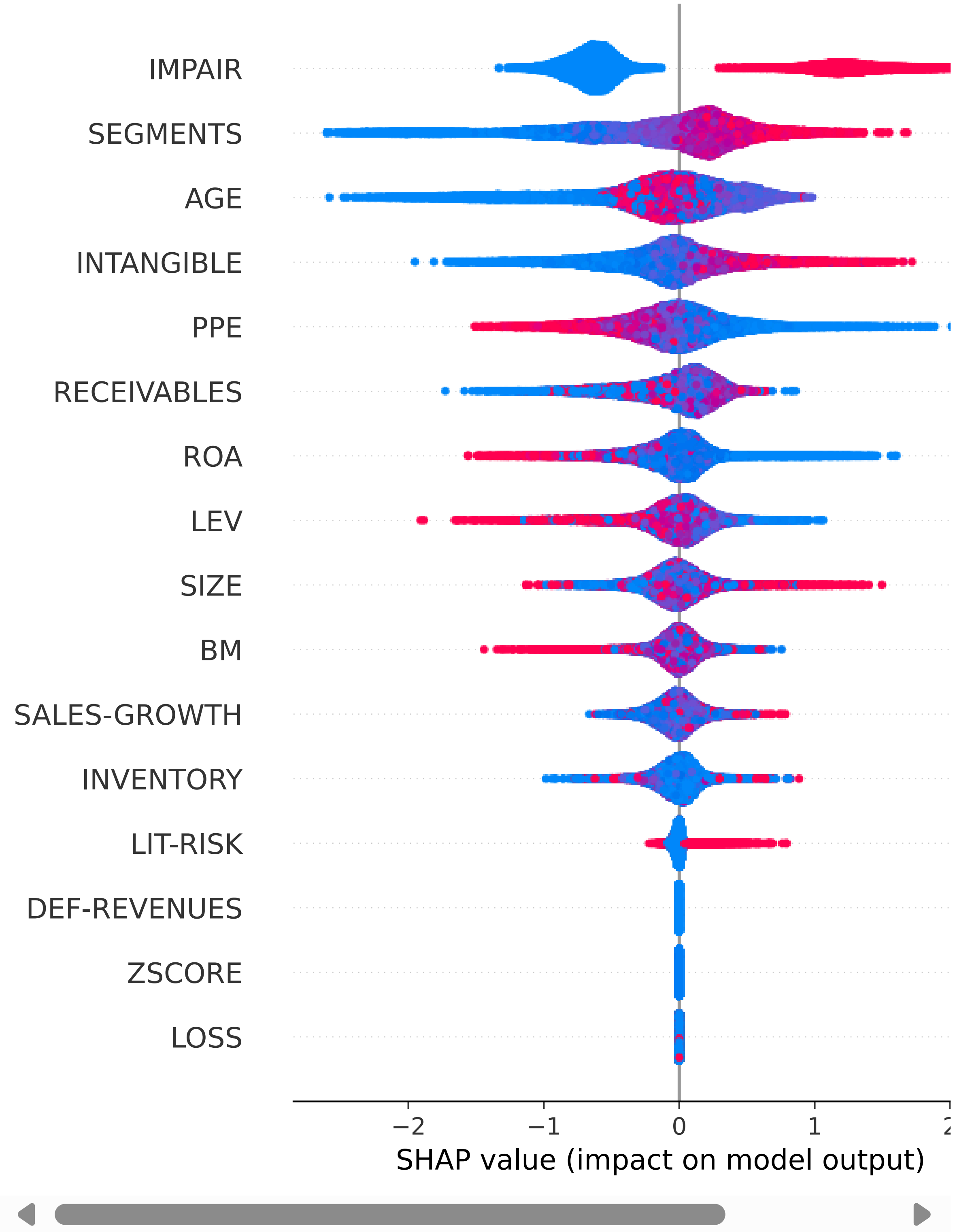
```
shap.summary_plot(shap_values[2], X)
shap.summary_plot(shap_values[3], X)
shap.summary_plot(shap_values[4], X)
```

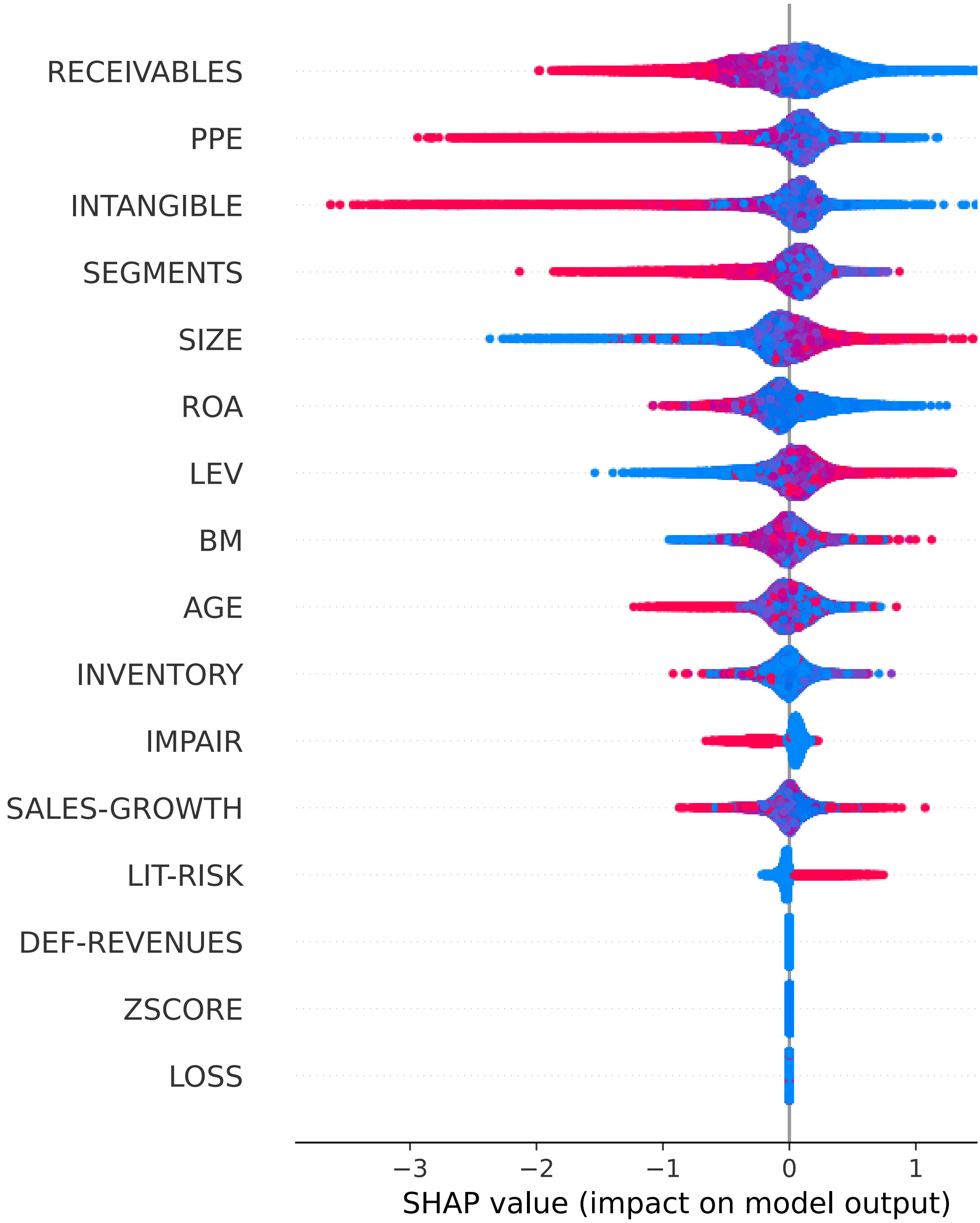
```
i=0, j=Revenue
i=1, j=other
i=2, j=Accounts-receivable
i=3, j=intangible-assets
i=4, j=inventories
```




RECEIVABLES
LEV
SIZE
AGE
ROA
INTANGIBLE
PPE
SEGMENTS
BM
SALES-GROWTH
INVENTORY
IMPAIR
LIT-RISK
DEF-REVENUES
ZSCORE
LOSS
−1
0
1
2
SHAP value (impact on model output)


RECEIVABLES
AGE
PPE
INTANGIBLE
SIZE
INVENTORY
ROA
BM
LEV
SALES-GROWTH
SEGMENTS
IMPAIR
LIT-RISK
DEF-REVENUES
ZSCORE
LOSS
−5
−4
−3
−2
−1
0
1
2
SHAP value (impact on model output)


IMPAIR
SEGMENTS
AGE
INTANGIBLE
PPE
RECEIVABLES
ROA
LEV
SIZE
BM
SALES-GROWTH
INVENTORY
LIT-RISK
DEF-REVENUES
ZSCORE
LOSS
−2
−1
0
1
SHAP value (impact on model output)



累积局部效应图

注：各个类别下的ALE图是选取当前类别下影响力排前三的x变量进行作图

In [89]:
```
Xdf=pd.DataFrame(X,columns=['SIZE', 'AGE', 'BM', 'SALES-GROWTH',
        'SEGMENTS', 'LEV', 'ROA', 'LOSS', 'ZSCORE', 'DEF-REVENUES',
        'RECEIVABLES', 'INVENTORY', 'PPE', 'INTANGIBLE', 'IMPAIR', 'LIT-RISK'])
Xdf.head()
```

Out[89]:

	SIZE	AGE	BM	SALES-GROWTH	SEGMENTS	LEV	ROA	LOSS	ZS
0	27.445504	25.0	0.182068	1.920584	4.276666	0.805367	2.175455	0.0	
1	27.784040	26.0	0.143285	1.660109	4.174387	0.839813	2.832663	0.0	
2	28.055360	27.0	0.151756	0.736779	4.304065	0.845856	3.762839	0.0	
3	28.179102	28.0	0.149493	0.701988	4.317488	0.843590	4.270946	0.0	
4	28.256519	29.0	0.188550	0.866897	4.369448	0.812835	4.457447	0.0	

label0

In [90]:

```
class clf_label0():
    def predict(df):
        return(gbclf.predict_proba(df)[:, 0])

gbclf.predict_proba(Xdf)
gbclf.predict_proba(Xdf)[:,0]
```

Out[90]:

```
array([[9.3479383e-01, 9.1893333e-01, 4.1087458e-04, 1.9299595e-03,
        9.5174748e-01],
       [9.6768260e-01, 8.4030080e-01, 5.3716119e-04, 9.6379849e-04,
        9.6060938e-01],
       [9.8562163e-01, 9.2296547e-01, 7.1569026e-04, 9.8641647e-04,
        9.6511412e-01],
       ...,
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03]], dtype=float32)
```

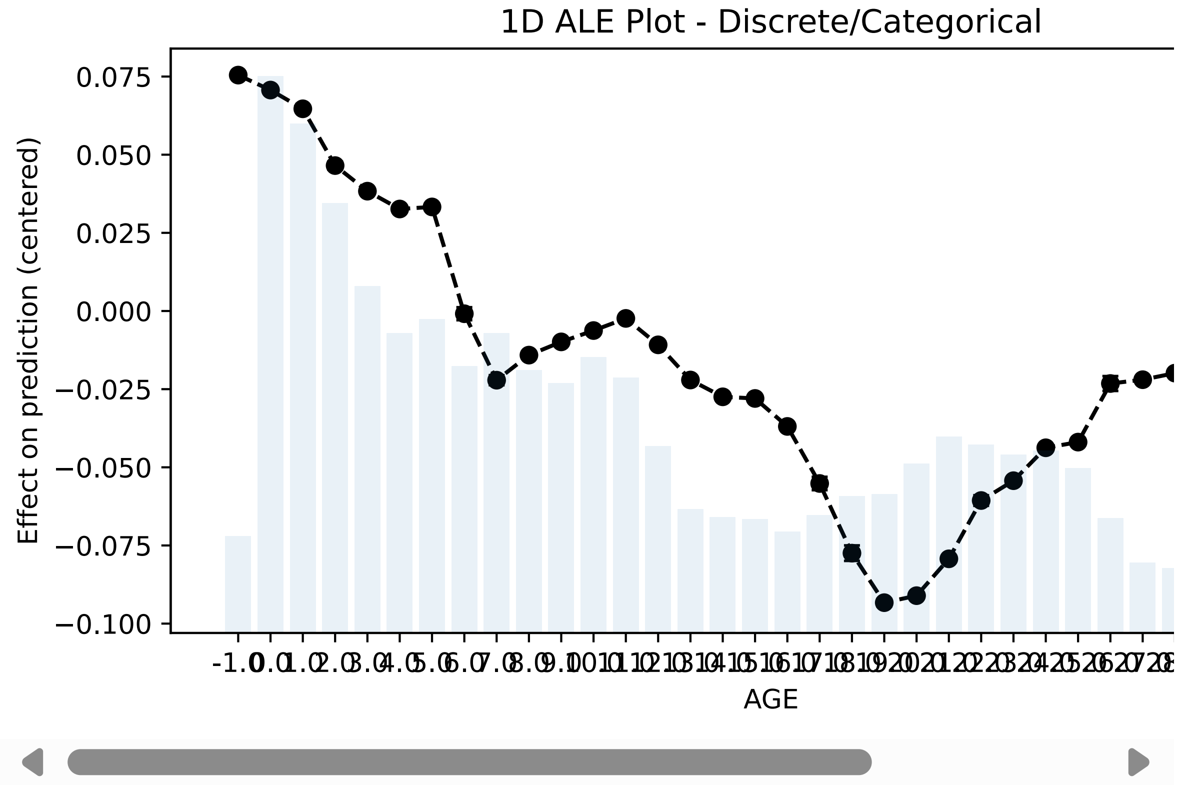
Out[90]:

```
array([0.93479383, 0.9676826 , 0.98562163, ..., 0.9823481 , 0.9823481 ,
       0.9823481 ], dtype=float32)
```

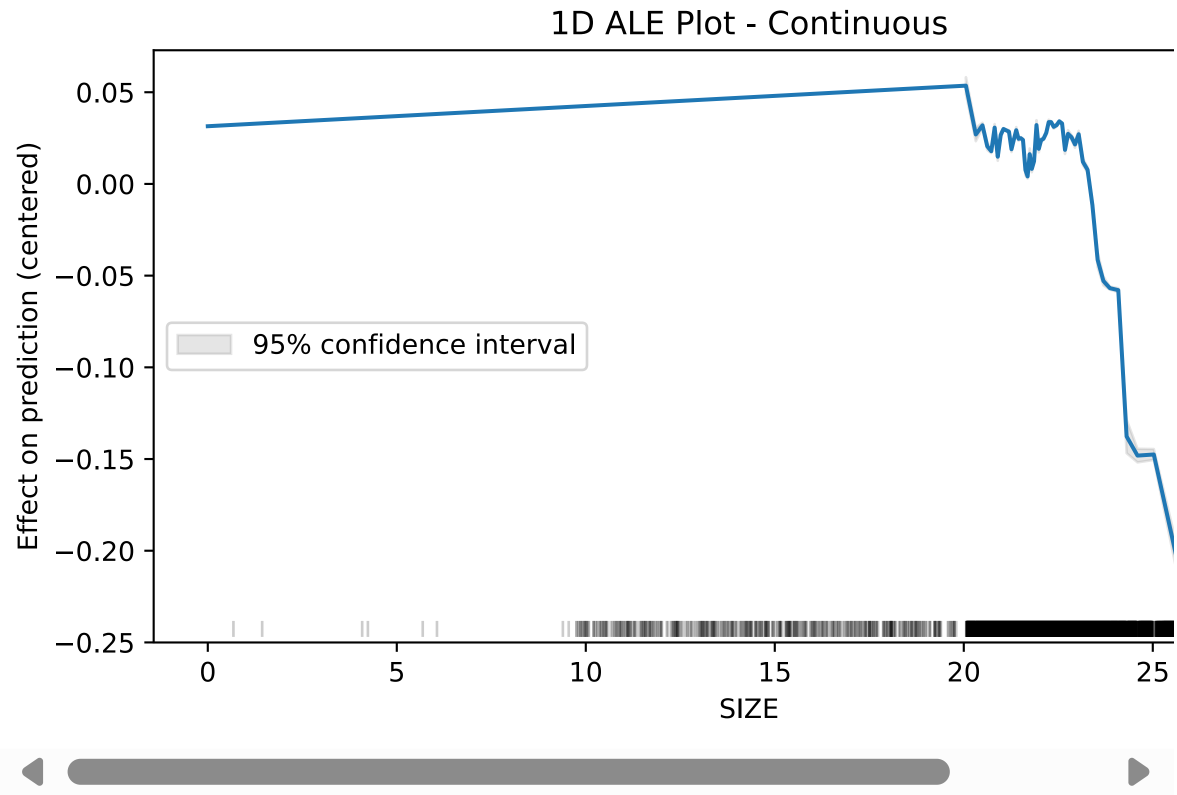
In [91]:

```
ale_eff = ale(X=Xdf, model=clf_label0, feature=["AGE"], grid_size=50, include_CI
ale_eff = ale(X=Xdf, model=clf_label0, feature=["SIZE"], grid_size=50, include_C
ale_eff = ale(X=Xdf, model=clf_label0, feature=["ROA"], grid_size=50, include_CI
```

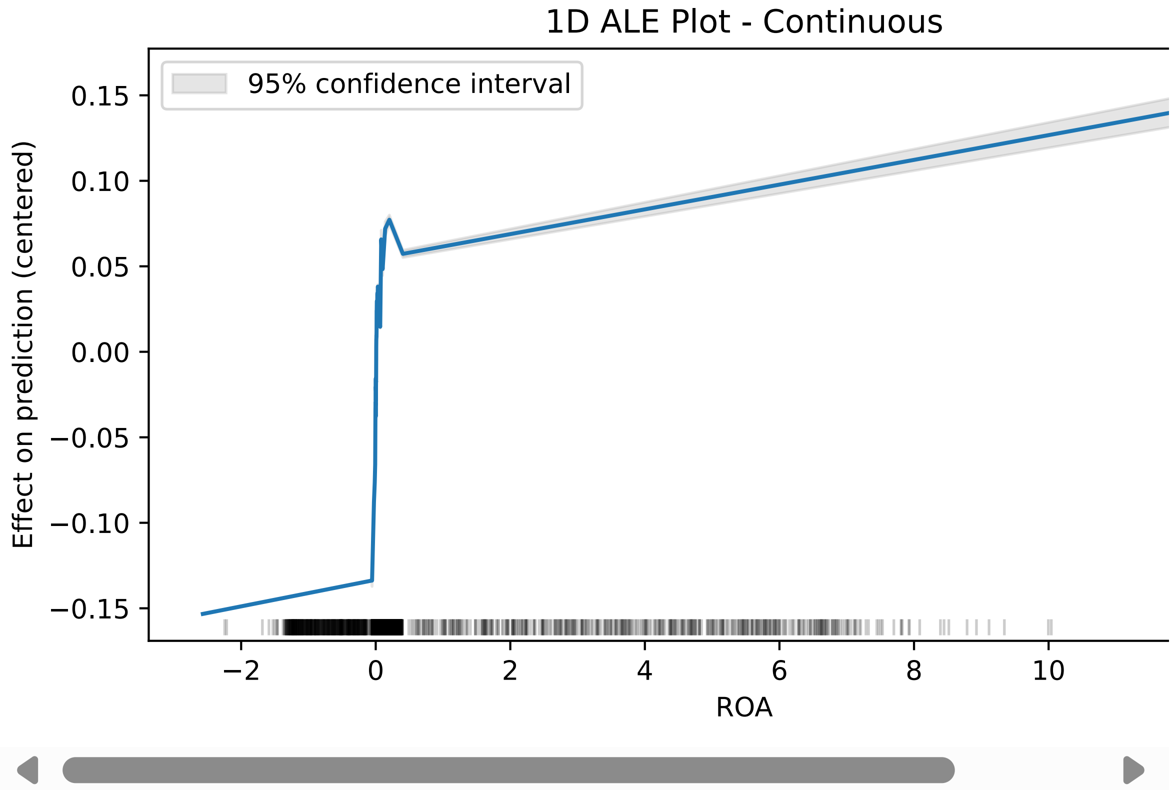
```
PyALE._ALE_generic:INFO: Discrete feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
```



1D ALE Plot - Discrete/Categorical
Effect on prediction (centered)
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
AGE



1D ALE Plot - Continuous
Effect on prediction (centered)
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
95% confidence interval
0
5
10
15
20
25
SIZE



label1

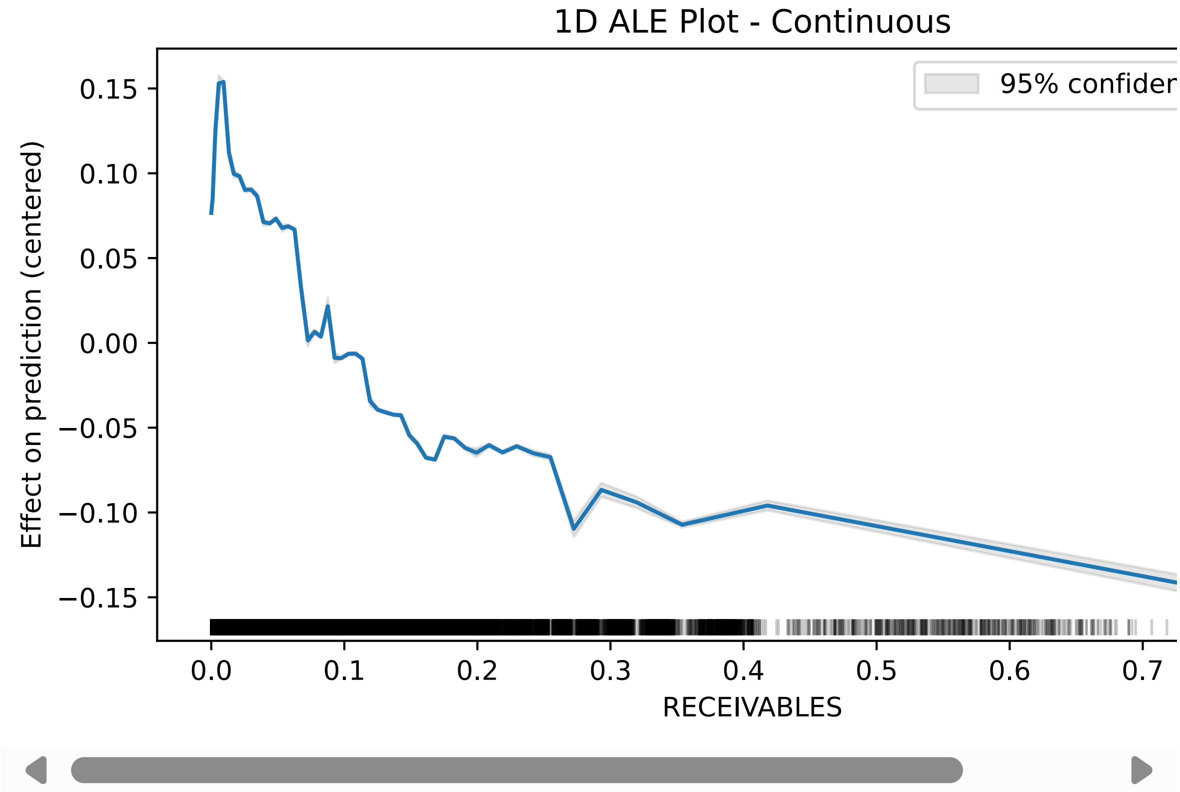
In [92]:
```
class clf_label1():
    def predict(df):
        return(gbclf.predict_proba(df)[:, 1])

gbclf.predict_proba(Xdf)
gbclf.predict_proba(Xdf)[:,1]
```

Out[92]:
```
array([[9.3479383e-01, 9.1893333e-01, 4.1087458e-04, 1.9299595e-03,
        9.5174748e-01],
       [9.6768260e-01, 8.4030080e-01, 5.3716119e-04, 9.6379849e-04,
        9.6060938e-01],
       [9.8562163e-01, 9.2296547e-01, 7.1569026e-04, 9.8641647e-04,
        9.6511412e-01],
       ...,
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03]], dtype=float32)
```

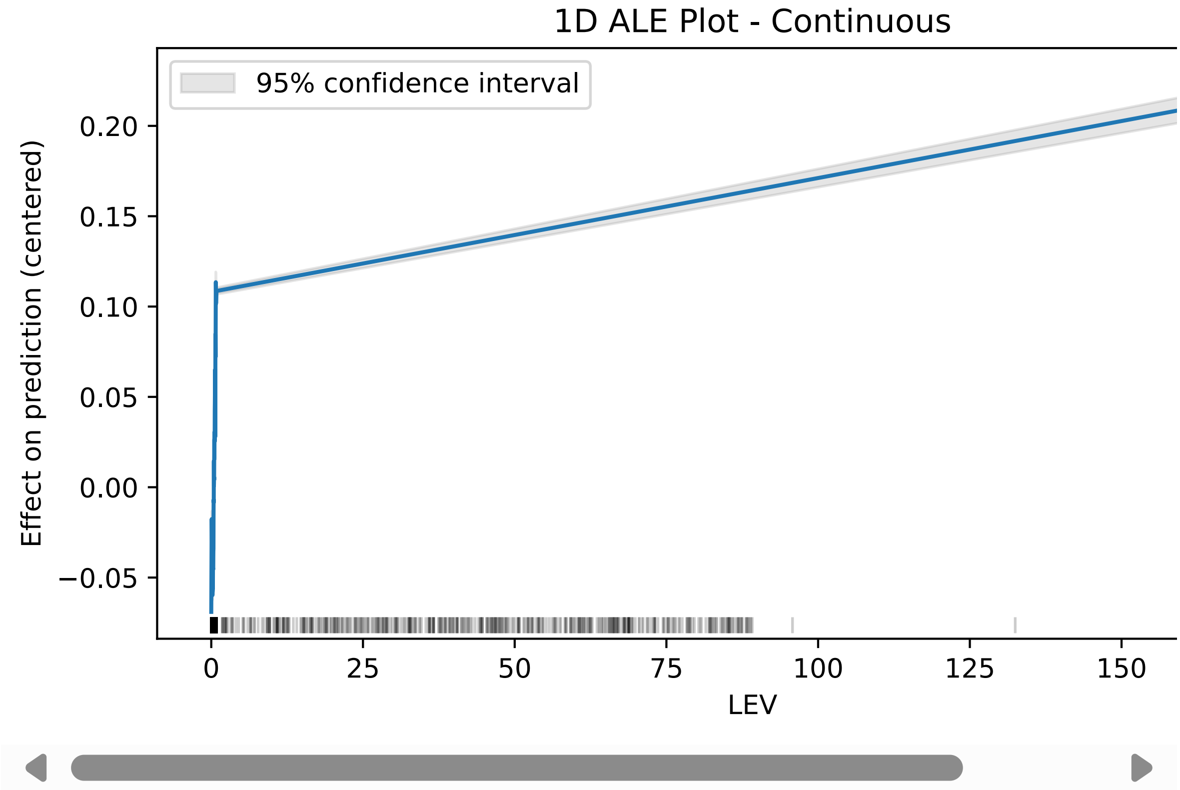
Out[92]:
```
array([0.91893333, 0.8403008 , 0.92296547, ..., 0.585852  , 0.585852  ,
       0.585852  ], dtype=float32)
```

In [93]:
```
ale_eff = ale(X=Xdf, model=clf_label1, feature=["RECEIVABLES"], grid_size=50, in
ale_eff = ale(X=Xdf, model=clf_label1, feature=["LEV"], grid_size=50, include_CI
ale_eff = ale(X=Xdf, model=clf_label1, feature=["SIZE"], grid_size=50, include_C
```

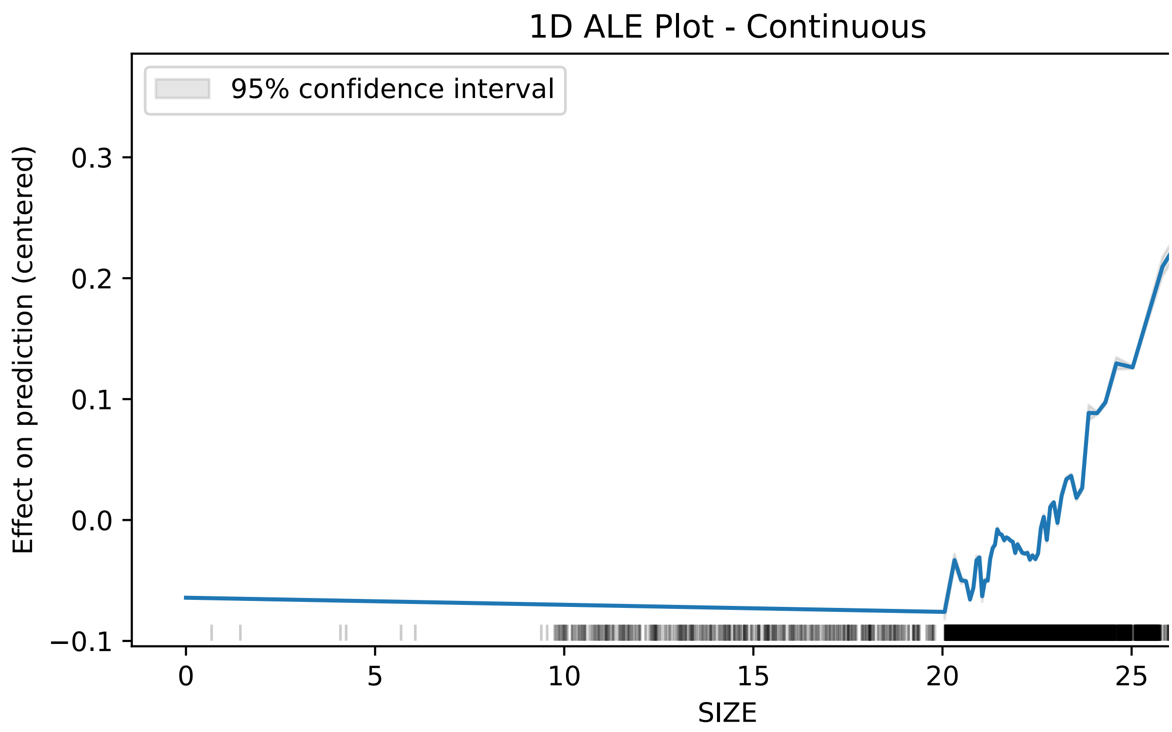
```
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
```



1D ALE Plot - Continuous
95% confider
Effect on prediction (centered)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
RECEIVABLES



1D ALE Plot - Continuous
95% confidence interval
Effect on prediction (centered)
0.20
0.15
0.10
0.05
0.00
−0.05
0
25
50
75
100
125
150
LEV



label2

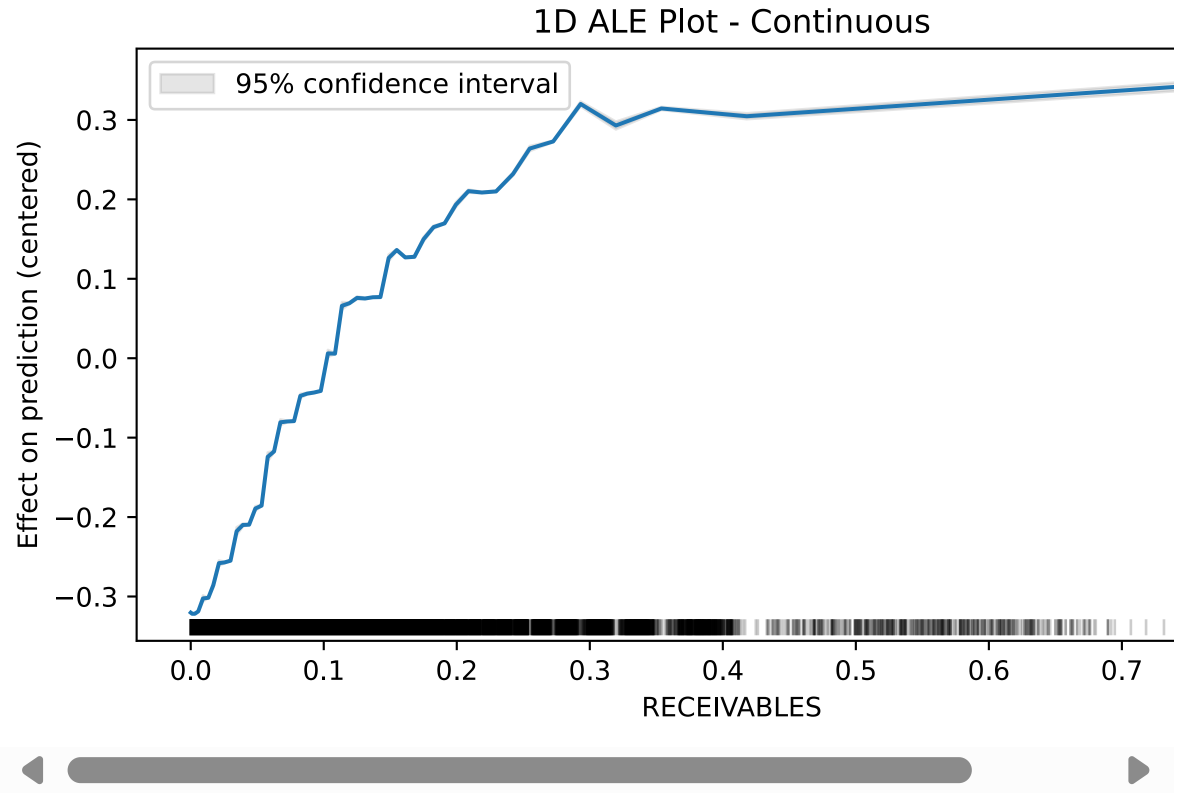
In [94]:
```
class clf_label2():
    def predict(df):
        return(gbclf.predict_proba(df)[:, 2])

gbclf.predict_proba(Xdf)
gbclf.predict_proba(Xdf)[:,2]
```

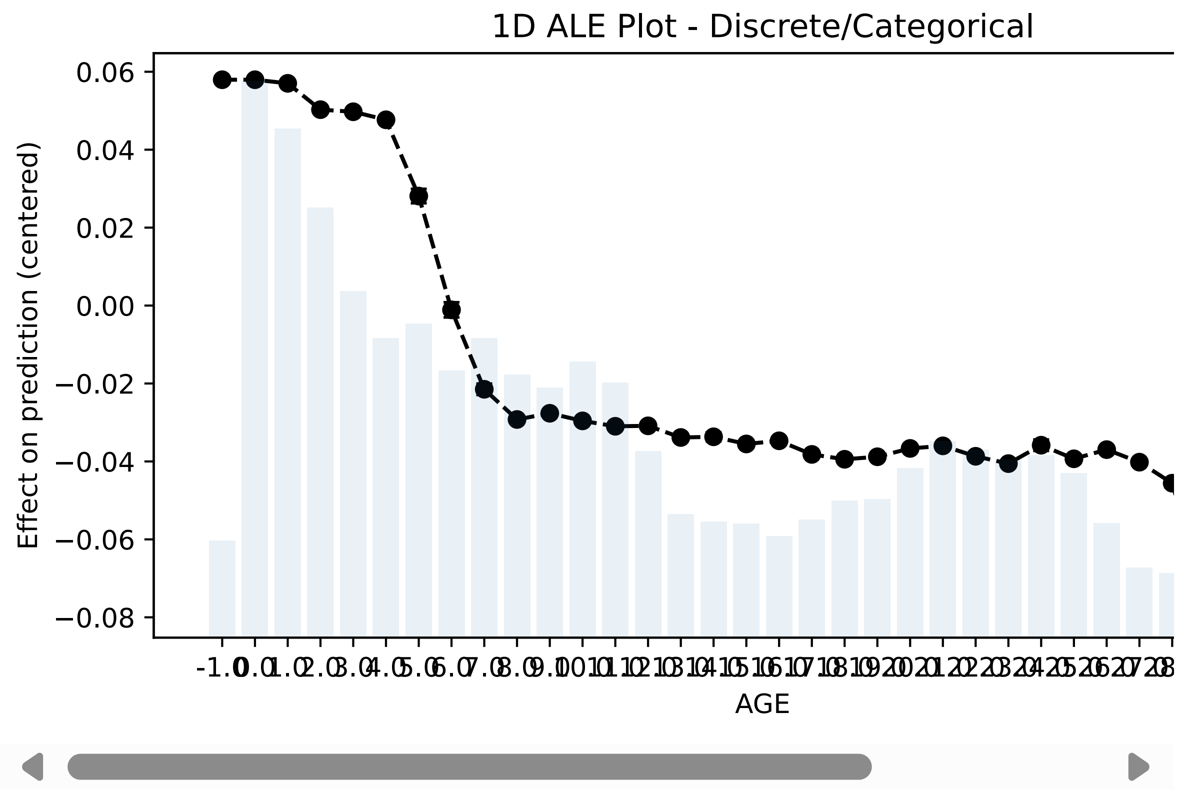
Out[94]:
```
array([[9.3479383e-01, 9.1893333e-01, 4.1087458e-04, 1.9299595e-03,
        9.5174748e-01],
       [9.6768260e-01, 8.4030080e-01, 5.3716119e-04, 9.6379849e-04,
        9.6060938e-01],
       [9.8562163e-01, 9.2296547e-01, 7.1569026e-04, 9.8641647e-04,
        9.6511412e-01],
       ...,
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03]], dtype=float32)
```

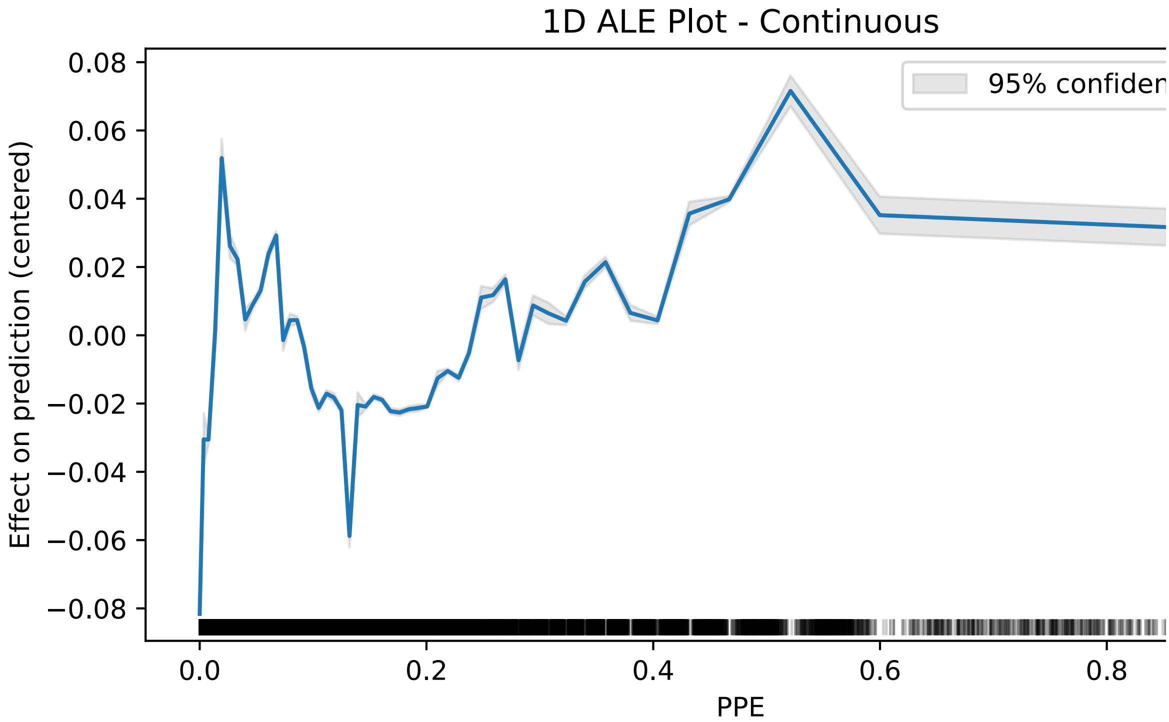
Out[94]:
```
array([4.1087458e-04, 5.3716119e-04, 7.1569026e-04, ..., 8.4301764e-01,
       8.4301764e-01, 8.4301764e-01], dtype=float32)
```

In [95]:
```
ale_eff = ale(X=Xdf, model=clf_label2, feature=["RECEIVABLES"], grid_size=50, ir
ale_eff = ale(X=Xdf, model=clf_label2, feature=["AGE"], grid_size=50, include_CI
ale_eff = ale(X=Xdf, model=clf_label2, feature=["PPE"], grid_size=50, include_CI
```

```
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Discrete feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
```


1D ALE Plot - Continuous
95% confidence interval
Effect on prediction (centered)
RECEIVABLES


1D ALE Plot - Discrete/Categorical
Effect on prediction (centered)
AGE



label3

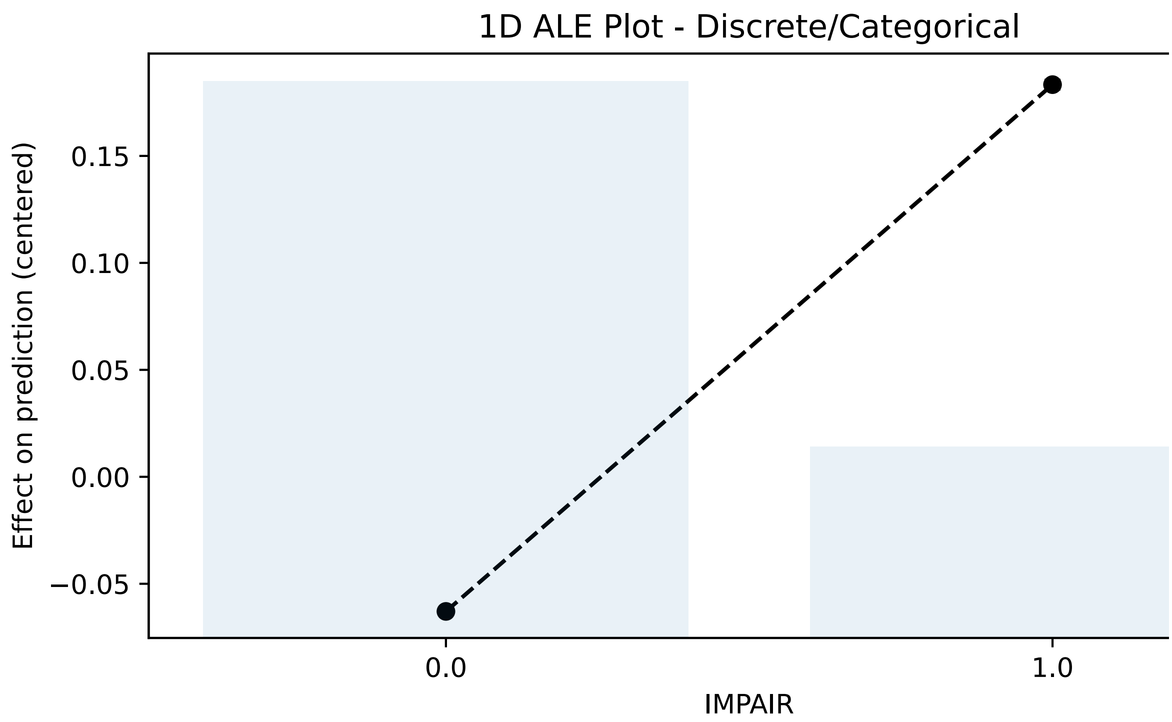
In [96]:
```
class clf_label3():
    def predict(df):
        return(gbclf.predict_proba(df)[:, 3])

gbclf.predict_proba(Xdf)
gbclf.predict_proba(Xdf)[:,3]
```

Out[96]:
```
array([[9.3479383e-01, 9.1893333e-01, 4.1087458e-04, 1.9299595e-03,
        9.5174748e-01],
       [9.6768260e-01, 8.4030080e-01, 5.3716119e-04, 9.6379849e-04,
        9.6060938e-01],
       [9.8562163e-01, 9.2296547e-01, 7.1569026e-04, 9.8641647e-04,
        9.6511412e-01],
       ...,
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03],
       [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
        6.5977490e-03]], dtype=float32)
```

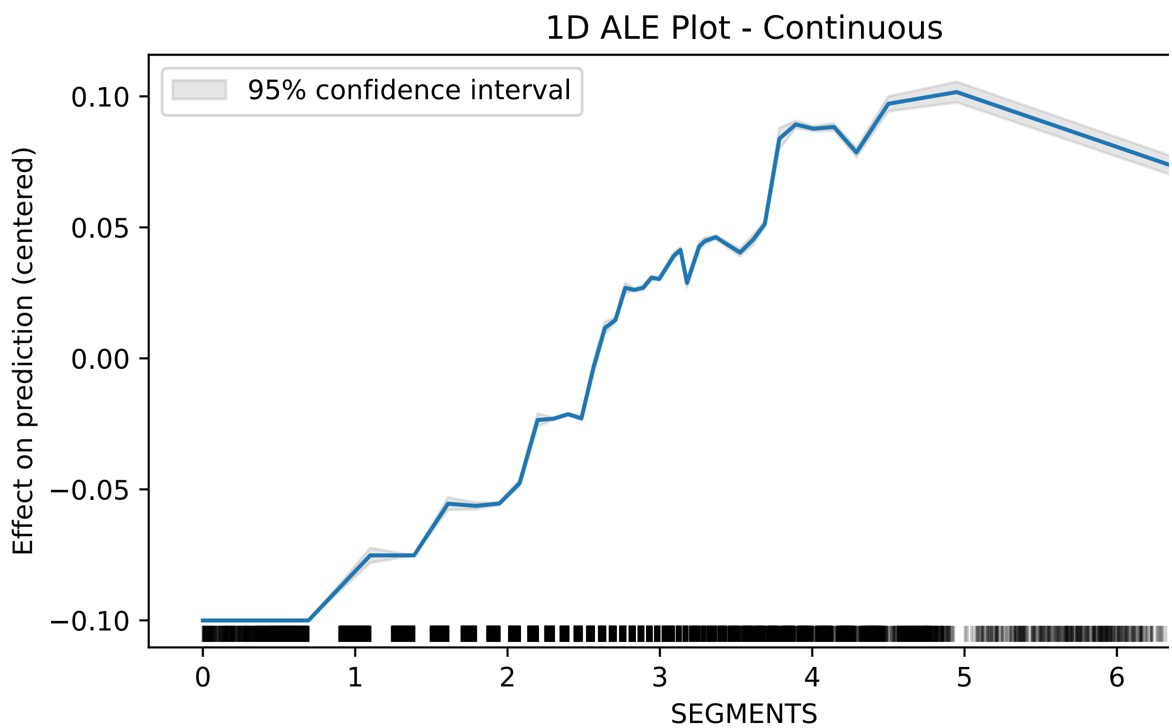
Out[96]:
```
array([0.00192996, 0.0009638 , 0.00098642, ..., 0.3034737 , 0.3034737 ,
       0.3034737 ], dtype=float32)
```

In [97]:
```
ale_eff = ale(X=Xdf, model=clf_label3, feature=['IMPAIR'], grid_size=50, include
ale_eff = ale(X=Xdf, model=clf_label3, feature=['SEGMENTS'], grid_size=50, inclu
ale_eff = ale(X=Xdf, model=clf_label3, feature=["AGE"], grid_size=50, include_CI
```

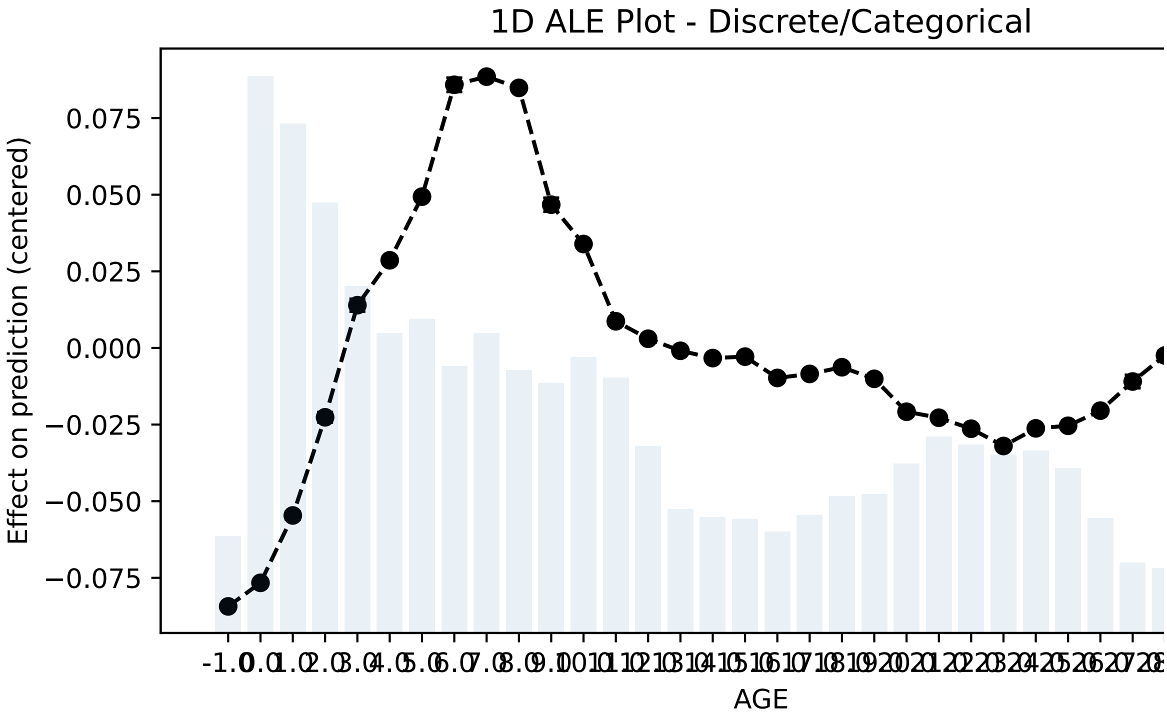
```
PyALE._ALE_generic:INFO: Discrete feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Discrete feature detected.
```



1D ALE Plot - Discrete/Categorical
Effect on prediction (centered)
0.15
0.10
0.05
0.00
−0.05
0.0
1.0
IMPAIR



1D ALE Plot - Continuous
95% confidence interval
Effect on prediction (centered)
0.10
0.05
0.00
−0.05
−0.10
0
1
2
3
4
5
6
SEGMENTS



label4

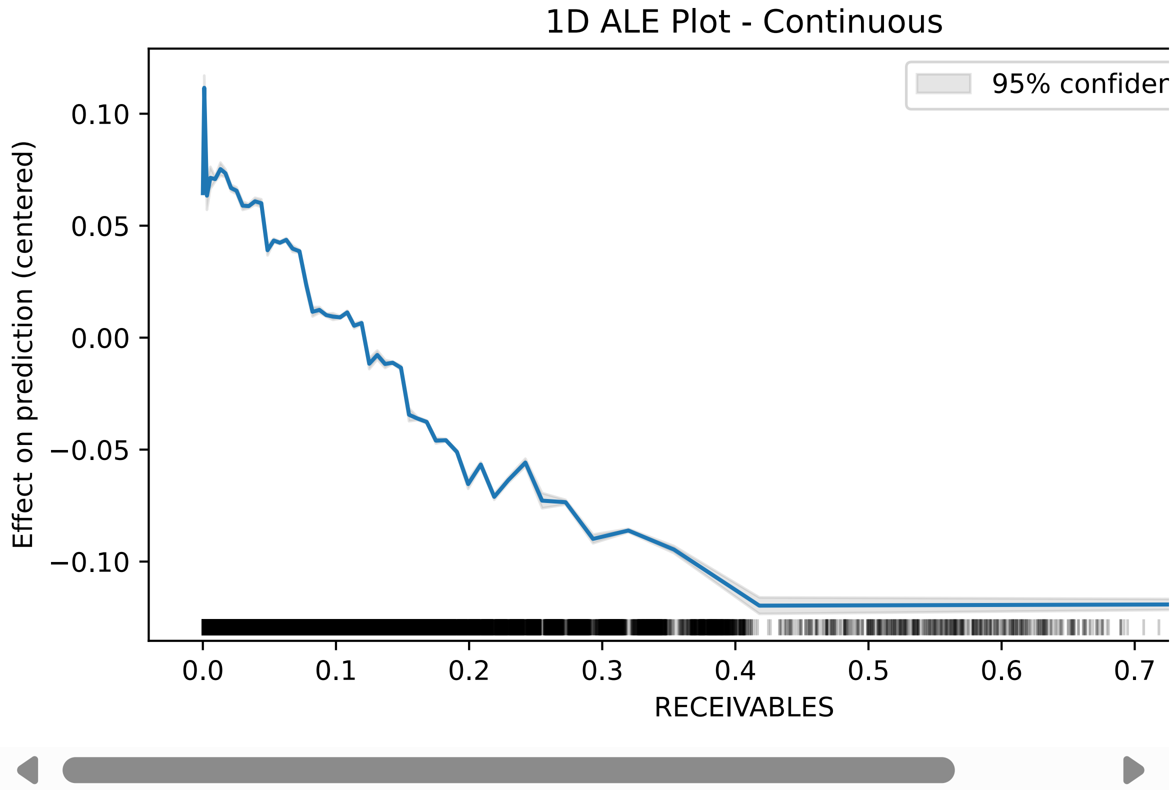
In [98]:
```
class clf_label4():
    def predict(df):
        return(gbclf.predict_proba(df)[:, 4])

gbclf.predict_proba(Xdf)
gbclf.predict_proba(Xdf)[:,4]
```

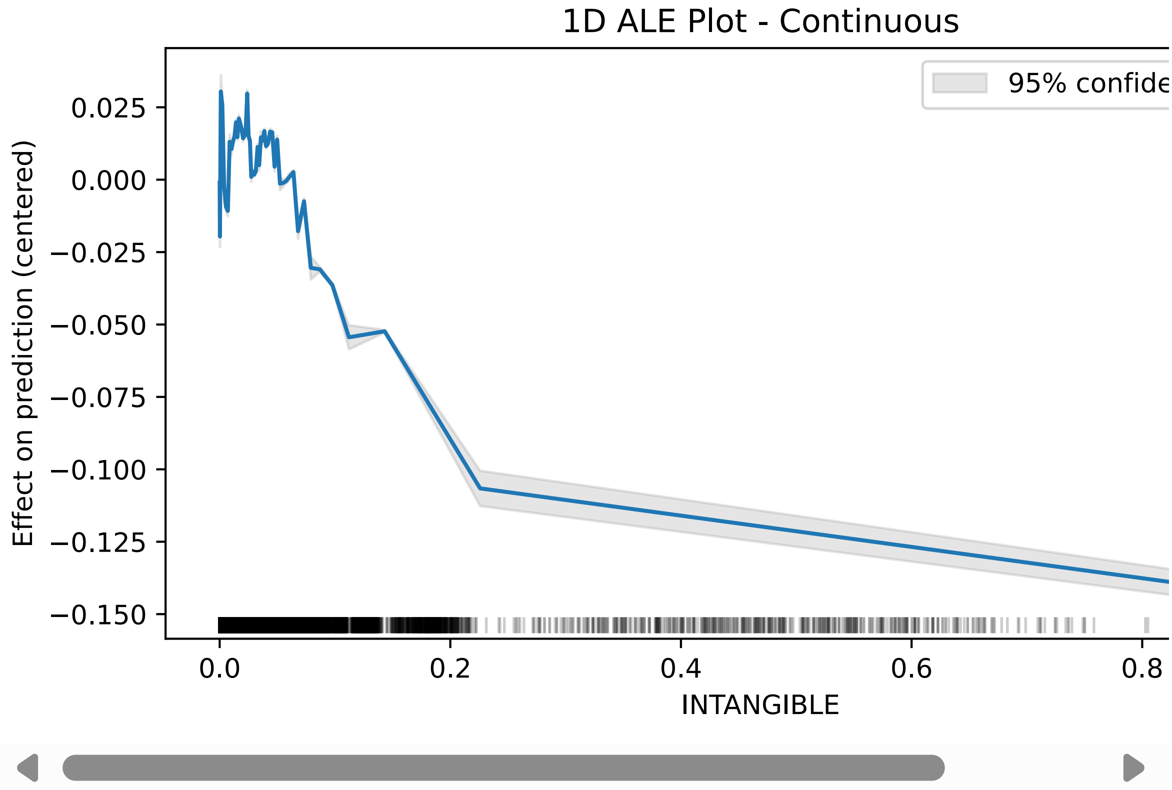
Out[98]:
```
array([[9.3479383e-01, 9.1893333e-01, 4.1087458e-04, 1.9299595e-03,
         9.5174748e-01],
        [9.6768260e-01, 8.4030080e-01, 5.3716119e-04, 9.6379849e-04,
         9.6060938e-01],
        [9.8562163e-01, 9.2296547e-01, 7.1569026e-04, 9.8641647e-04,
         9.6511412e-01],
        ...,
        [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
         6.5977490e-03],
        [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
         6.5977490e-03],
        [9.8234808e-01, 5.8585203e-01, 8.4301764e-01, 3.0347371e-01,
         6.5977490e-03]], dtype=float32)
```

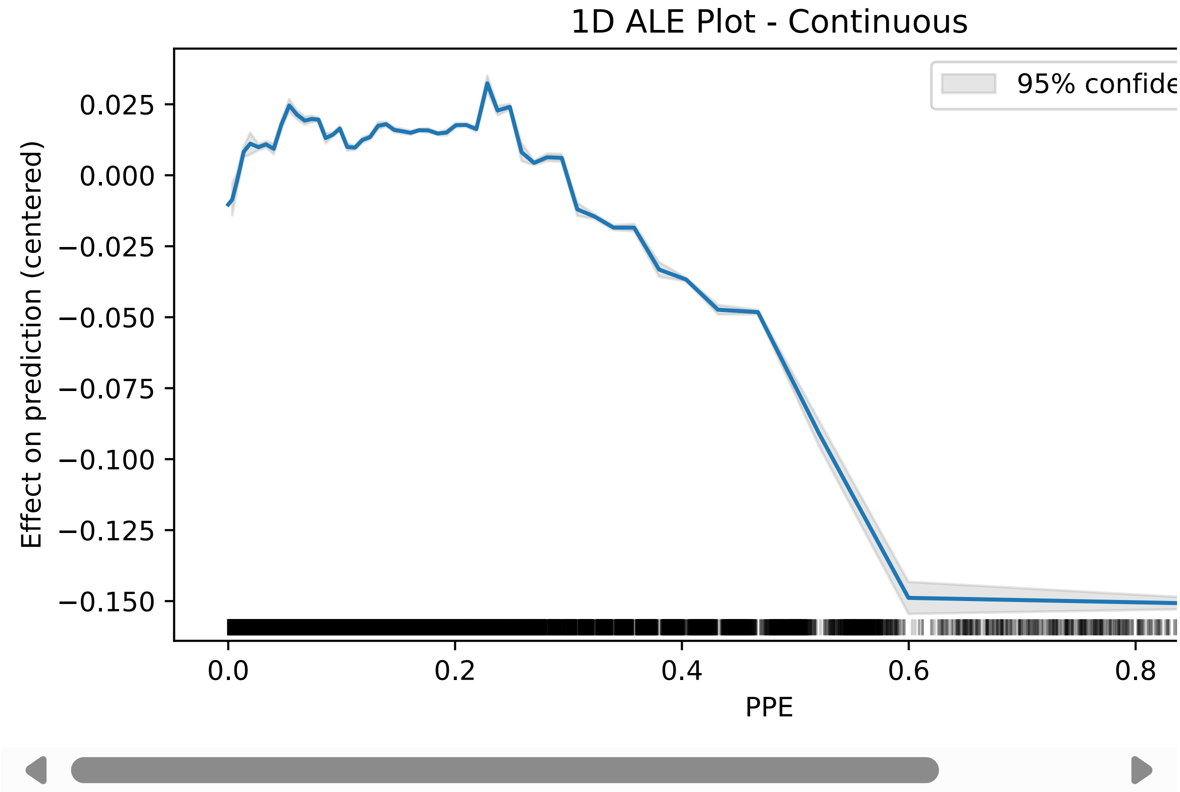
Out[98]:
```
array([0.9517475 , 0.9606094 , 0.9651141 , ..., 0.00659775, 0.00659775,
       0.00659775], dtype=float32)
```

In [99]:
```
ale_eff = ale(X=Xdf, model=clf_label4, feature=['RECEIVABLES'], grid_size=50, in
ale_eff = ale(X=Xdf, model=clf_label4, feature=['INTANGIBLE'], grid_size=50, inc
ale_eff = ale(X=Xdf, model=clf_label4, feature=["PPE"], grid_size=50, include_CI
```

```
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
PyALE._ALE_generic:INFO: Continuous feature detected.
```


1D ALE Plot - Continuous
95% confider
Effect on prediction (centered)
0.10
0.05
0.00
−0.05
−0.10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
RECEIVABLES


1D ALE Plot - Continuous
95% confide
Effect on prediction (centered)
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.125
−0.150
0.0
0.2
0.4
0.6
0.8
INTANGIBLE


1D ALE Plot - Continuous
95% confide
Effect on prediction (centered)
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.125
−0.150
0.0
0.2
0.4
0.6
0.8
PPE