05 Evaluating Classification Supervised Learning Model Quality

Nandi Christmas

In [53]:
```python
import pandas as pd
import numpy as np
df = pd.read_csv(
    filepath_or_buffer = 'D:\\loan_data.csv',
    engine = 'pyarrow',
    dtype = {
        'person_education_2_moderate': str,
        'person_education_3_high': str,
        'person_home_ownership_3_high': str,
        'loan_intent_2_moderate': str,
        'loan_intent_3_high' : str,
    }
)
```

In [54]:
```python
df
```

Out[54]:

	loan_status	partition	person_education_2_moderate	person_education_3_high	person_home_ownership_3_high	loan_intent_2_moderate	loan_intent_3_high
0	1	test	0	0	1	1	0
1	0	test	1	0	0	0	0
2	1	train	1	0	0	0	1
3	1	train	0	1	1	0	1
4	1	test	0	0	1	0	1
...	...	...	...	...	...	...	...
44995	1	test	0	0	1	0	1
44996	1	validation	0	0	1	1	0
44997	1	train	0	0	1	0	1
44998	1	train	0	1	1	0	0
44999	1	train	1	0	1	0	1

45000 rows × 7 columns

List of prepared predictor variables

In [55]:
```python
list_predictors = [
        'person_education_2_moderate',
        'person_education_3_high',
        'person_home_ownership_3_high',
        'loan_intent_2_moderate',
        'loan_intent_3_high',
]
```

Subset the data to the training target and predictors

In [56]:
```python
X_train = df.loc[df['partition'] == 'train'][list_predictors]
y_train = df.loc[df['partition'] == 'train']['loan_status']
```

Use LASSO for feature selection

In [57]:
```python
from sklearn.linear_model import LogisticRegression
LogisticRegression_l1 = LogisticRegression(
    penalty='l1',
    random_state=823,
    solver='liblinear'
).fit(
    X = X_train,
    y = y_train
)
```

In [58]:
```python
[
    predictor for predictor, is_non_zero in zip(list_predictors, LogisticRegression_l1.coef_.flatten() != 0) if is_non_zero
]
list_predictors = [
        'person_education_2_moderate',
        'person_education_3_high',
        'person_home_ownership_3_high',
        'loan_intent_2_moderate',
        'loan_intent_3_high',
]
```

Fit a logistic regression model

In [59]:
```python
from sklearn.linear_model import LogisticRegression
LogisticRegression_None = LogisticRegression(
    penalty=None,
    random_state=823,
    #solver='liblinear'
).fit(
    X = X_train[list_predictors],
    y = y_train
)
LogisticRegression_None
```

Out[59]:

LogisticRegression

LogisticRegression(penalty=None, random_state=823)

Get fitted values from the logistic regression model

In [60]:
```python
df['predict'] = LogisticRegression_None.predict(
    X = df[list_predictors]
)
```

In [61]:
```python
df['predict_proba'] = pd.DataFrame(LogisticRegression_None.predict_proba(
    X = df[list_predictors]
))[1]
df[['loan_status','predict','predict_proba']]
```

Out[61]:

	loan_status	predict	predict_proba
0	1	0	0.346188
1	0	0	0.074189
2	1	0	0.144433
3	1	0	0.413999
4	1	0	0.417970
...	...	...	...
44995	1	0	0.417970
44996	1	0	0.346188
44997	1	0	0.417970
44998	1	0	0.251134
44999	1	0	0.398035

45000 rows × 3 columns

Common classification supervised learning metrics

In [62]:
```python
df['partition'].value_counts()
```

Out[62]:
```
partition
test          15060
train         15028
validation    14912
Name: count, dtype: int64
```

Classification report

In [63]:
```python
from sklearn.metrics import classification_report
[
    print(classification_report(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
    )) for partition in ['train','validation','test']
]
```

```
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
              precision    recall  f1-score   support

           0       0.77      1.00      0.87     11596
           1       0.00      0.00      0.00      3432

    accuracy                           0.77     15028
   macro avg       0.39      0.50      0.44     15028
weighted avg       0.60      0.77      0.67     15028

d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
              precision    recall  f1-score   support

           0       0.78      1.00      0.88     11691
           1       0.00      0.00      0.00      3221

    accuracy                           0.78     14912
   macro avg       0.39      0.50      0.44     14912
weighted avg       0.61      0.78      0.69     14912

d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
              precision    recall  f1-score   support

           0       0.78      1.00      0.87     11713
           1       0.00      0.00      0.00      3347

    accuracy                           0.78     15060
   macro avg       0.39      0.50      0.44     15060
weighted avg       0.60      0.78      0.68     15060

d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Out[63]: [None, None, None]

Confusion Matrix

In [64]:
```python
from sklearn.metrics import confusion_matrix
[
    confusion_matrix(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
    ) for partition in ['train','validation','test']
]
```

Out[64]:
```
[array([[11596,     0],
        [ 3432,     0]]),
 array([[11691,     0],
        [ 3221,     0]]),
 array([[11713,     0],
        [ 3347,     0]])]
```

In [65]:
```python
df_metric = pd.DataFrame({
    'TrainTest' : ['train','validation','test']
}).set_index('TrainTest')
```

Accuracy

In [66]:
```python
from sklearn.metrics import accuracy_score
df_metric['accuracy_score'] = [
    accuracy_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'],
        y_pred = df.loc[(df['partition'] == partition)]['predict']
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[66]:

TrainTest	accuracy_score
train	0.771626
validation	0.783999
test	0.777756

Balanced Accuracy

In [67]:
```python
from sklearn.metrics import balanced_accuracy_score
df_metric['balanced_accuracy_score'] = [
    balanced_accuracy_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'],
        y_pred = df.loc[(df['partition'] == partition)]['predict']
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[67]:

TrainTest	accuracy_score	balanced_accuracy_score
train	0.771626	0.5
validation	0.783999	0.5
test	0.777756	0.5

Precision, recall, F-measure and support for each class.

In [68]:
```python
from sklearn.metrics import precision_recall_fscore_support
[
    (precision_recall_fscore_support(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(float),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(float),
    )) for partition in ['train','validation','test']
]
```

```
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Out[68]:
```
[(array([0.7716263, 0.       ]),
  array([1., 0.]),
  array([0.87109375, 0.        ]),
  array([11596,  3432])),
 (array([0.78399946, 0.        ]),
  array([1., 0.]),
  array([0.87892343, 0.        ]),
  array([11691,  3221])),
 (array([0.77775564, 0.        ]),
  array([1., 0.]),
  array([0.87498599, 0.        ]),
  array([11713,  3347]))]
```

Sensitivity Recall

In [69]:
```python
from sklearn.metrics import recall_score
df_metric['recall_score'] = [
    recall_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
        zero_division=0,
        pos_label='1'
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[69]:

TrainTest	accuracy_score	balanced_accuracy_score	recall_score
train	0.771626	0.5	0.0
validation	0.783999	0.5	0.0
test	0.777756	0.5	0.0

Precision

In [70]:
```python
from sklearn.metrics import precision_score
df_metric['precision_score'] = [
    precision_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
        zero_division=0,
        pos_label='1'
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[70]:

TrainTest	accuracy_score	balanced_accuracy_score	recall_score	precision_score
train	0.771626	0.5	0.0	0.0
validation	0.783999	0.5	0.0	0.0
test	0.777756	0.5	0.0	0.0

F-1 Score

In [71]:
```python
from sklearn.metrics import f1_score
df_metric['f1_score'] = [
    f1_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
        zero_division=0,
        pos_label='1'
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[71]:

TrainTest	accuracy_score	balanced_accuracy_score	recall_score	precision_score	f1_score
train	0.771626	0.5	0.0	0.0	0.0
validation	0.783999	0.5	0.0	0.0	0.0
test	0.777756	0.5	0.0	0.0	0.0

Binary classification positive and negative likelihood ratios

In [72]:
```python
from sklearn.metrics import class_likelihood_ratios
[
    class_likelihood_ratios(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_pred = df.loc[(df['partition'] == partition)]['predict'].astype(str),
    ) for partition in ['train','validation','test']
]
```

```
d:\python\Lib\site-packages\sklearn\utils\_param_validation.py:213: UserWarning: no samples predicted for the positive class
  return func(*args, **kwargs)
d:\python\Lib\site-packages\sklearn\utils\_param_validation.py:213: UserWarning: no samples predicted for the positive class
  return func(*args, **kwargs)
d:\python\Lib\site-packages\sklearn\utils\_param_validation.py:213: UserWarning: no samples predicted for the positive class
  return func(*args, **kwargs)
```

Out[72]: [(nan, np.float64(1.0)), (nan, np.float64(1.0)), (nan, np.float64(1.0))]

Average precision score

In [73]:
```python
from sklearn.metrics import average_precision_score
df_metric['average_precision_score'] = [
    average_precision_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(float),
        y_score = df.loc[(df['partition'] == partition)]['predict_proba'],
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[73]:

TrainTest	accuracy_score	balanced_accuracy_score	recall_score	precision_score	f1_score	average_precision_score
train	0.771626	0.5	0.0	0.0	0.0	0.349912
validation	0.783999	0.5	0.0	0.0	0.0	0.323336
test	0.777756	0.5	0.0	0.0	0.0	0.333081

precision-recall pairs for different probability thresholds

In [74]:
```python
from sklearn.metrics import precision_recall_curve
[
    precision_recall_curve(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(float),
        probas_pred = df.loc[(df['partition'] == partition)]['predict_proba'],
    ) for partition in ['train','validation','test']
]
```

```
d:\python\Lib\site-packages\sklearn\metrics\_ranking.py:993: FutureWarning: probas_pred was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_score`` instead.
  warnings.warn(
d:\python\Lib\site-packages\sklearn\metrics\_ranking.py:993: FutureWarning: probas_pred was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_score`` instead.
  warnings.warn(
d:\python\Lib\site-packages\sklearn\metrics\_ranking.py:993: FutureWarning: probas_pred was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_score`` instead.
  warnings.warn(
```

Out[74]:
```
[(array([0.2283737 , 0.23695286, 0.24845276, 0.26729354, 0.27593895,
         0.28691945, 0.30349044, 0.30984825, 0.31788869, 0.33329108,
         0.34200481, 0.35120297, 0.37121668, 0.37564361, 0.37833595,
         0.375     , 0.36175047, 0.39480801, 1.        ]),
  array([1.        , 0.98426573, 0.97086247, 0.95250583, 0.93764569,
         0.92162005, 0.89685315, 0.86509324, 0.82051282, 0.76602564,
         0.70483683, 0.63374126, 0.53962704, 0.48892774, 0.42132867,
         0.32255245, 0.23892774, 0.13432401, 0.        ]),
  array([0.07418851, 0.07886564, 0.08006127, 0.1106938 , 0.11738057,
         0.11908461, 0.14443341, 0.15280811, 0.15493618, 0.23889015,
         0.25113412, 0.25422061, 0.32774843, 0.34249727, 0.34618755,
         0.39803536, 0.41399928, 0.41797023])),
 (array([0.21600054, 0.22381322, 0.2348661 , 0.25310623, 0.26035912,
         0.2696845 , 0.28578584, 0.29140484, 0.30180902, 0.31678214,
         0.32510347, 0.3348032 , 0.35296491, 0.3525713 , 0.35095027,
         0.34684385, 0.3451171 , 0.32839314, 1.        ]),
  array([1.        , 0.9850978 , 0.97205837, 0.95498292, 0.93635517,
         0.91555418, 0.88574977, 0.84942564, 0.81837939, 0.76653213,
         0.70723378, 0.63644831, 0.54330953, 0.48742627, 0.41850357,
         0.32412294, 0.24247128, 0.13070475, 0.        ]),
  array([0.07418851, 0.07886564, 0.08006127, 0.1106938 , 0.11738057,
         0.11908461, 0.14443341, 0.15280811, 0.15493618, 0.23889015,
         0.25113412, 0.25422061, 0.32774843, 0.34249727, 0.34618755,
         0.39803536, 0.41399928, 0.41797023])),
 (array([0.22224436, 0.23115683, 0.24275687, 0.26074873, 0.26883283,
         0.27852805, 0.29461196, 0.30056711, 0.30954808, 0.32190911,
         0.32700956, 0.33729904, 0.36166401, 0.36152827, 0.35917114,
         0.36479843, 0.35494107, 0.34956795, 1.        ]),
  array([1.        , 0.9868539 , 0.9713176 , 0.95309232, 0.93934867,
         0.91813564, 0.89034957, 0.85509411, 0.81655214, 0.75769346,
         0.69524948, 0.62683   , 0.54287422, 0.48909471, 0.41948013,
         0.3325366 , 0.23394084, 0.13295488, 0.        ]),
  array([0.07418851, 0.07886564, 0.08006127, 0.1106938 , 0.11738057,
         0.11908461, 0.14443341, 0.15280811, 0.15493618, 0.23889015,
         0.25113412, 0.25422061, 0.32774843, 0.34249727, 0.34618755,
         0.39803536, 0.41399928, 0.41797023]))]
```

Brier score loss (MSE)

In [75]:
```python
from sklearn.metrics import brier_score_loss
df_metric['brier_score_loss'] = [
    brier_score_loss(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(float),
        y_prob = df.loc[(df['partition'] == partition)]['predict_proba'],
    ) for partition in ['train','validation','test']
]
df_metric
```

```
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:3239: FutureWarning: y_prob was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_proba`` instead.
  warnings.warn(
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:3239: FutureWarning: y_prob was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_proba`` instead.
  warnings.warn(
d:\python\Lib\site-packages\sklearn\metrics\_classification.py:3239: FutureWarning: y_prob was deprecated in version 1.5 and will be removed in 1.7.Please use ``y_proba`` instead.
  warnings.warn(
```

Out[75]:

TrainTest	accuracy_score	balanced_accuracy_score	recall_score	precision_score	f1_score	average_precision_score	brier_score_loss
train	0.771626	0.5	0.0	0.0	0.0	0.349912	0.161875
validation	0.783999	0.5	0.0	0.0	0.0	0.323336	0.156974
test	0.777756	0.5	0.0	0.0	0.0	0.333081	0.159835

Area under the Curve (AUC)

In [76]:
```python
from sklearn.metrics import roc_auc_score
df_metric['roc_auc_score'] = [
    roc_auc_score(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(str),
        y_score = df.loc[(df['partition'] == partition)]['predict_proba'].astype(float),
    ) for partition in ['train','validation','test']
]
df_metric
```

Out[76]:

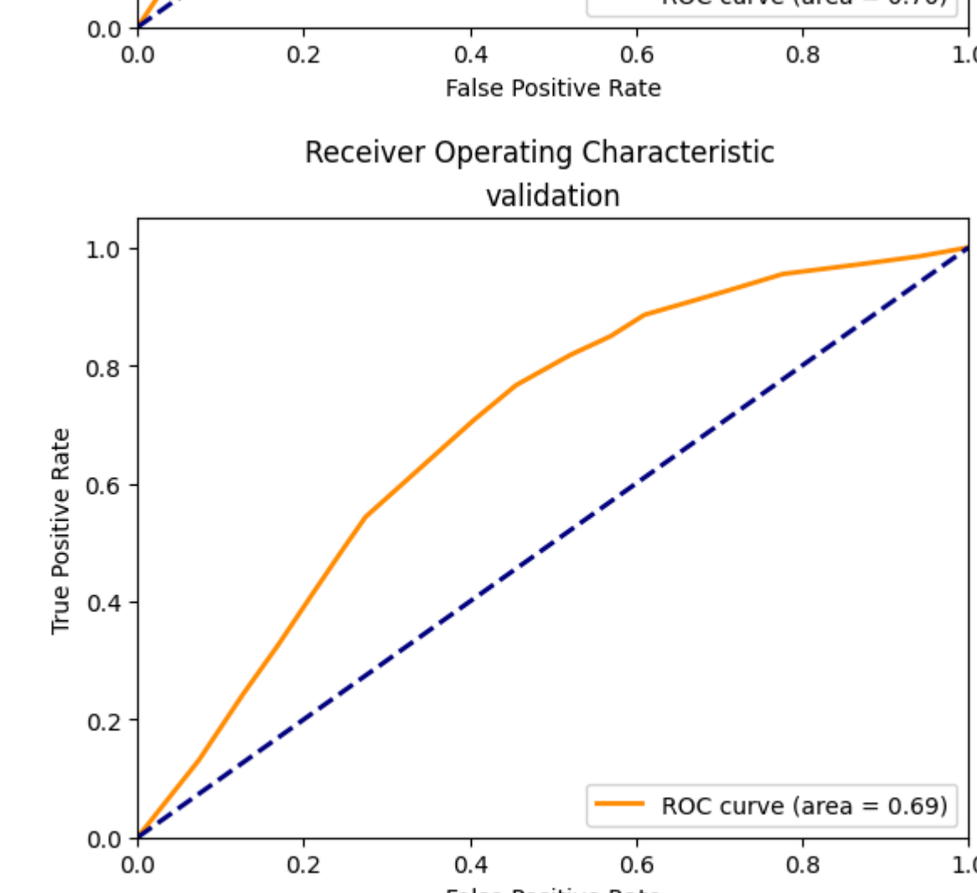
TrainTest	accuracy_score	balanced_accuracy_score	recall_score	precision_score	f1_score	average_precision_score	brier_score_loss	roc_auc_score
train	0.771626	0.5	0.0	0.0	0.0	0.349912	0.161875	0.696780
validation	0.783999	0.5	0.0	0.0	0.0	0.323336	0.156974	0.690306
test	0.777756	0.5	0.0	0.0	0.0	0.333081	0.159835	0.690289

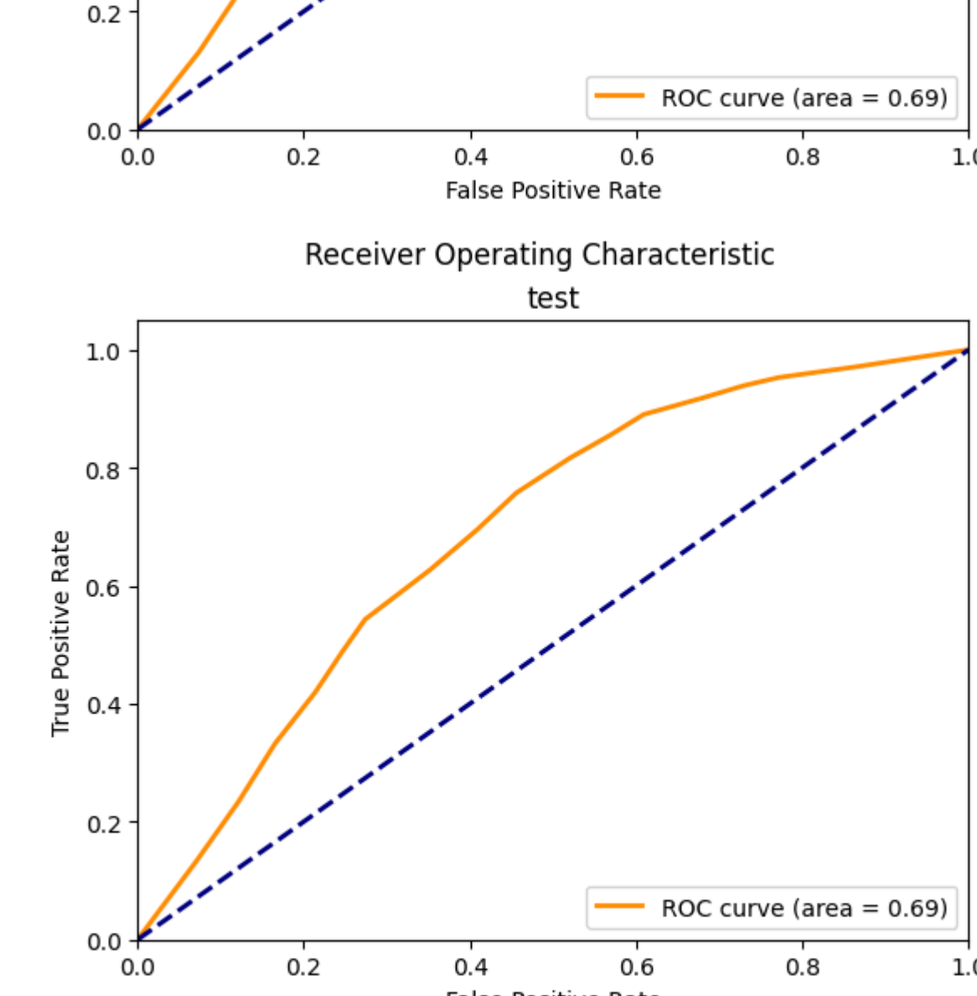
Visualize the Receiver Operating Characteristic curve

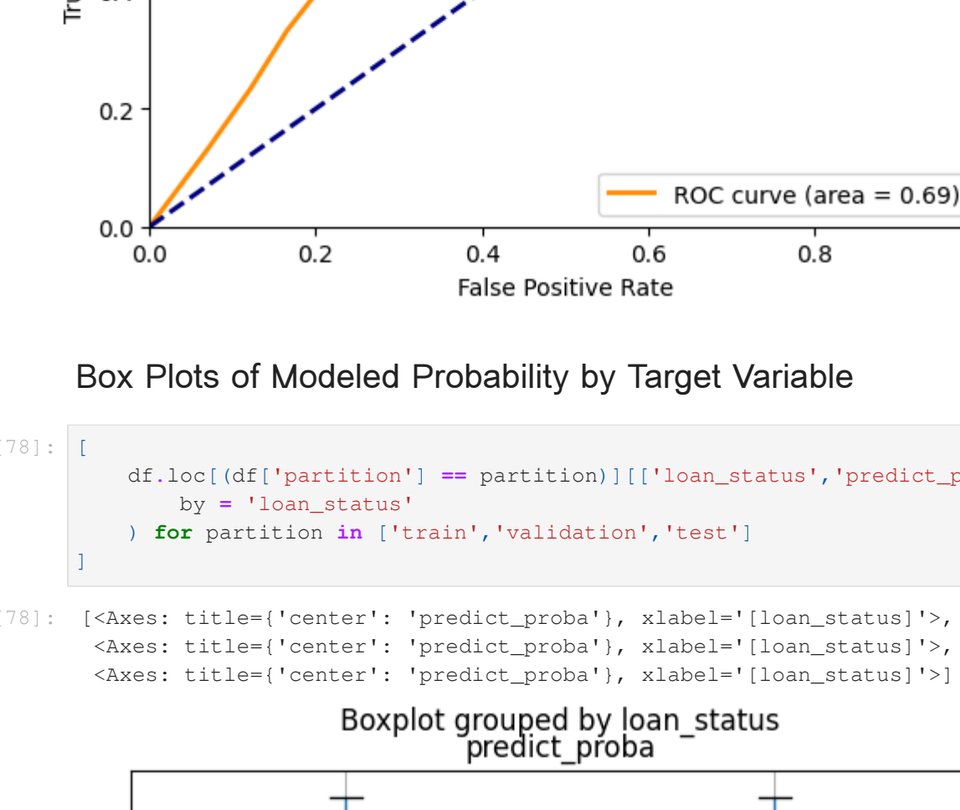
In [77]:
```python
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve,auc

for partition in ['train','validation','test']:
    # Compute ROC curve and ROC area for each class
    fpr, tpr, thresholds = roc_curve(
        y_true = df.loc[(df['partition'] == partition)]['loan_status'].astype(float),
        y_score = df.loc[(df['partition'] == partition)]['predict_proba'].astype(float)
    )
    roc_auc = auc(fpr, tpr)

    # Plotting
    plt.figure()
    lw = 2  # Line width
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.suptitle('Receiver Operating Characteristic')
    plt.title(partition)
    plt.legend(loc="lower right")
    plt.show()
```







Box Plots of Modeled Probability by Target Variable

In [78]:
```python
[
    df.loc[(df['partition'] == partition)][['loan_status','predict_proba']].boxplot(
        by = 'loan_status'
    ) for partition in ['train','validation','test']
]
```

Out[78]:
```
[<Axes: title={'center': 'predict_proba'}, xlabel='[loan_status]'>,
 <Axes: title={'center': 'predict_proba'}, xlabel='[loan_status]'>,
 <Axes: title={'center': 'predict_proba'}, xlabel='[loan_status]'>]
```