```python
import pickle

def load_and_print_artifacts_dict(path):
    artifacts_dict = pickle.load(open(path, "rb"))

    print("Target encoder mapping:")
    print([ac for ac in artifacts_dict["encoder"].mapping])

    print("Columns to train:")
    print([ac for ac in artifacts_dict["columns_to_score"]])

if __name__ == "__main__":

load_and_print_artifacts_dict("./Artifacts/artifacts_dict_file.pkl")
```

```
Target encoder mapping:
['City', 'State', 'Bank', 'BankState', 'RevLineCr', 'LowDoc',
'NewExist']
Columns to train:
['City_trg', 'State_trg', 'Zip_trg', 'Bank_trg', 'BankState_trg',
'NAICS_trg', 'NoEmp_trg', 'NewExist_trg', 'CreateJob_trg',
'RetainedJob_trg', 'FranchiseCode_trg', 'UrbanRural_trg',
'RevLineCr_trg', 'LowDoc_trg', 'DisbursementGross_trg',
'BalanceGross_trg', 'GrAppv_trg', 'SBA_Appv_trg', 'Zip', 'NAICS',
'NoEmp', 'CreateJob', 'RetainedJob', 'FranchiseCode', 'UrbanRural',
'DisbursementGross', 'BalanceGross', 'GrAppv', 'SBA_Appv',
'Log_DisbursementGross', 'Log_NoEmp', 'Log_GrAppv', 'Log_SBA_Appv',
'Log_BalanceGross', 'Disbursement_Bins', 'Loan_Efficiency',
'Guarantee_Ratio', 'Loan_Guarantee_Interaction',
'Disbursement_Squared']
```

```python
from matplotlib import pyplot as plt
import shap
from sklearn.inspection import permutation_importance
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
import lightgbm as lgb
import warnings
import numpy as np
import pandas as pd
warnings.filterwarnings("ignore", category=Warning)

def scoring(data):
    """
    Function to score input dataset.

    Input: dataset in Pandas DataFrame format
    Output: Python list of labels in the same order as input records

    Flow:
        - Load artifacts
        - Transform dataset
        - Score dataset
        - Return labels

    """
    artifacts_dict_file =
open("D:/Work/Gre/UTD/Courses/Fall/MIS6341/Softwares/Python/ml-fall-
2023/Project2/artifacts/artifacts_dict_file.pkl", "rb")
    artifacts_dict = pickle.load(file=artifacts_dict_file)
    artifacts_dict_file.close()
    best_classifier = artifacts_dict["best_classifier"]
    encoder = artifacts_dict["encoder"]
    scaler = artifacts_dict["scaler"]
    threshold = artifacts_dict["optimal_threshold"]
    numerical_columns = artifacts_dict["numerical_columns"]
    cat_cols = artifacts_dict["cat_cols"]
    columns_to_score = artifacts_dict["columns_to_score"]

    for i in data['RevLineCr']:
        if i not in ['Y','N']:
            data['RevLineCr'].replace(i,'N',inplace=True)
    for i in data['LowDoc']:
        if i not in ['Y','N']:
            data['LowDoc'].replace(i,'N',inplace=True)
    for i in data['NewExist']:
        if i not in [1,2]:
            data['NewExist'].replace(i,None,inplace=True)

    for column in cat_cols:
        data[column]=data[column].fillna(data[column].mode()[0])

    data_encoded = encoder.transform(data)
    data_encoded = data_encoded.add_suffix('_trg')
    data_encoded = pd.concat([data_encoded, data], axis=1)
    for column in cat_cols:
        data_encoded[column + "_trg"].fillna(data_encoded[column +
"_trg"].mean(), inplace=True)
    data_encoded.drop(columns=cat_cols, inplace=True)

    data_encoded['Log_DisbursementGross'] =
np.log1p(data_encoded['DisbursementGross'])
    data_encoded['Log_NoEmp'] = np.log1p(data_encoded['NoEmp'])
    data_encoded['Log_GrAppv'] = np.log1p(data_encoded['GrAppv'])
    data_encoded['Log_SBA_Appv'] = np.log1p(data_encoded['SBA_Appv'])
    data_encoded['Log_BalanceGross'] =
np.log1p(data_encoded['BalanceGross'])

    data_encoded['Disbursement_Bins'] =
pd.cut(data_encoded['DisbursementGross'],
                                                bins=[-np.inf, 50000,
150000, np.inf],
                                                labels=['Low',
'Medium', 'High'])

    data_encoded['Loan_Efficiency'] =
data_encoded['DisbursementGross'] / (data_encoded['CreateJob'] +
data_encoded['RetainedJob'] + 1)  # Adding 1 to avoid division by zero

    data_encoded['Guarantee_Ratio'] = data_encoded['SBA_Appv'] /
data_encoded['GrAppv']

    data_encoded['Loan_Guarantee_Interaction'] =
data_encoded['SBA_Appv'] * data_encoded['GrAppv']

    data_encoded['Disbursement_Squared'] =
data_encoded['DisbursementGross'] ** 2

    data_encoded[numerical_columns] =
scaler.transform(data_encoded[numerical_columns])

    y_prob =
best_classifier.predict_proba(data_encoded[columns_to_score])
    y_pred = (y_prob[:,0] < threshold).astype(int)
    d = {
        "index" : data_encoded.index,
        "label" : y_pred,
        "probability_0": y_prob[:,0],
        "probability_1": y_prob[:,1]
    }



    return pd.DataFrame(d)

import pandas as pd
df2 =
pd.read_csv("D:/Work/Gre/UTD/Courses/Fall/MIS6341/Softwares/Python/ml-
fall-2023/Project2/SBA_loans_project_2_holdout_students_valid.csv")

print(scoring(df2))
```

```
[LightGBM] [Warning] min_data_in_leaf is set=300, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=300
[LightGBM] [Warning] feature_fraction is set=0.9, colsample_bytree=1.0
will be ignored. Current value: feature_fraction=0.9
[LightGBM] [Warning] lambda_l1 is set=9.408025110972025, reg_alpha=0.0
will be ignored. Current value: lambda_l1=9.408025110972025
[LightGBM] [Warning] lambda_l2 is set=3.9690665922792114e-08,
reg_lambda=0.0 will be ignored. Current value:
lambda_l2=3.9690665922792114e-08
```

```
[LightGBM] [Warning] bagging_fraction is set=1.0, subsample=1.0 will
be ignored. Current value: bagging_fraction=1.0
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
       index  label  probability_0  probability_1
0          0      0       0.849864       0.150136
1          1      1       0.410310       0.589690
2          2      0       0.999752       0.000248
3          3      0       0.694401       0.305599
4          4      0       0.689322       0.310678
...      ...    ...            ...            ...
98904  98904      0       0.697117       0.302883
98905  98905      1       0.178386       0.821614
98906  98906      0       0.868871       0.131129
98907  98907      0       0.751461       0.248539
98908  98908      1       0.597262       0.402738

[98909 rows x 4 columns]
```