MLP Report (Week 7)

This week we examined on multiple months prediction, and apply entropy evaluation with base "e". There are two ways for calculating entropy:

```
# 1. sample-based: -ln() on every sample and then average
# Range: (0, ln(num_classes))
def entropy(y_pred_proba, y_test):
    true_probs = np.sum(y_pred_proba * y_test, axis=1)
    log_probs = np.empty_like(true_probs)
    for i, p in enumerate(true_probs):
        if p == 0:
            print(f"Sample {i}: True class probability is 0, setting log to
NaN")
            log_probs[i] = np.nan
        else:
            log_probs[i] = np.log(p)

    entropy = -np.nanmean(log_probs)
    return entropy
```

Within classes from **0 to 6+**, entropy ranges from **0 to 2.079**.

```
# 2. class-based: -ln() on every class and then average
# similar to PTP calculation
def entropy(y_pred_proba, y_test):
    mean_entropy_class = []
    for i in range(y_pred_proba.shape[1]):
        rows_i = y_test[:, i] == 1
        if np.sum(rows_i) > 0:
            entropy_i = np.mean(-np.log(y_pred_proba[rows_i, i]))
        else:
            entropy_i = np.nan
        print(f"Entropy for class {i}: {entropy_i}")
        mean_entropy_class.append(entropy_i)

    mean_entropy = np.nanmean(mean_entropy_class)
    return mean_entropy
```

The results below are all based on predicting deliquency status after 3 months.

Iteration Method 1

***propagating the transition probabilities*
Under this method, two different approches are applied. In the following algorithm, either pass original `pred_proba` to `input`, or encode into one-hot form to align with training data.

```
# iterate to predict status after n months
def predict_n_months(mlp, n, input):
    for _ in range(n):
        pred_proba = mlp.predict_proba(input)
```

```
        # choose one following method
        # 1. remain probability form
        input = pred_proba

        # 2. encode to one-hot
        input_labels = np.argmax(pred_proba, axis=1)
        input = np.zeros_like(pred_proba)
        input[np.arange(len(input)), input_labels] = 1

    return pred_proba

y_pred_proba = predict_n_months(mlp, MONTH_AHEAD, X_test)
```

Generally, the latter achieves better result.

1. remain probability form

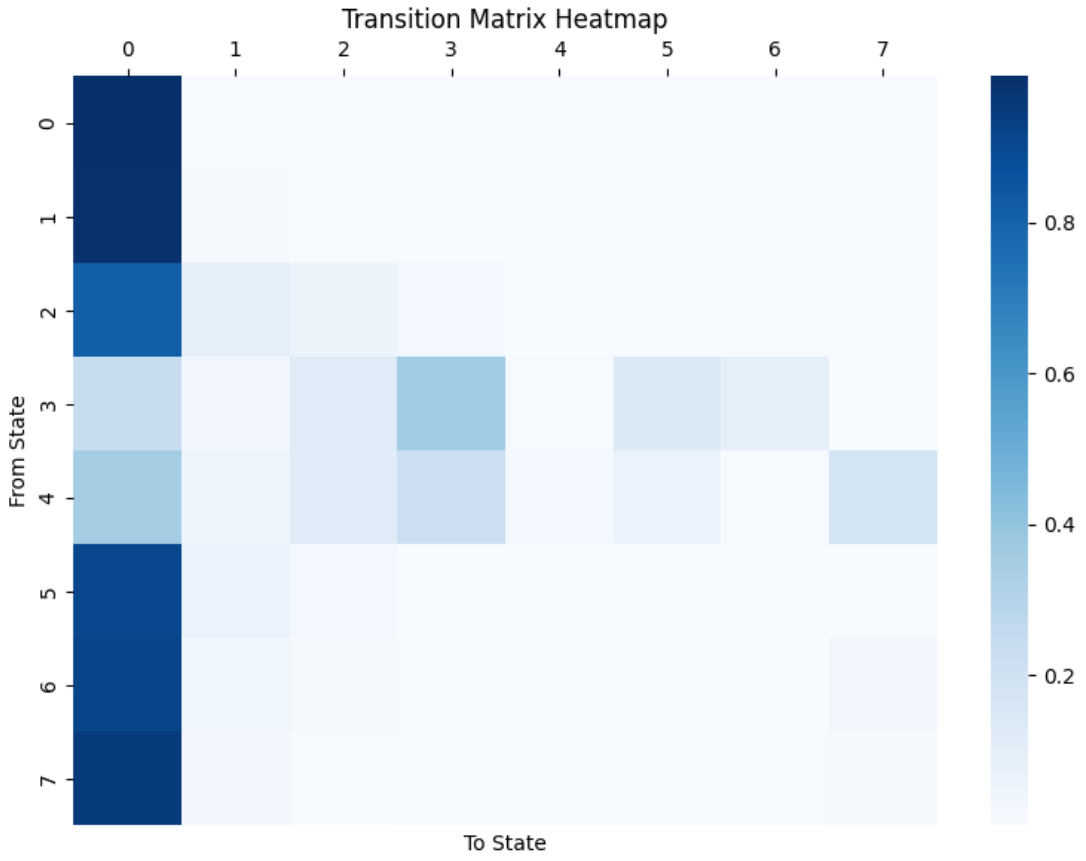
average probability = 0.12584506748065927
entropy = 0.16520659321309436
brier score = 0.037273658636576584
adjusted brier score = 0.0027203744763043556
entropy_class = 6.800633458961423

```
Entropy for class 0: 0.006389342406215621
Entropy for class 1: 4.951031872679091
Entropy for class 2: 8.672714212642067
Entropy for class 3: 6.75143950568151
Entropy for class 4: 6.0477956920122
Entropy for class 5: 10.762551816907925
Entropy for class 6: 6.497255739518178
Entropy for class 7: 10.715889489844193
```

With `Credit Score`:
average probability = 0.13524271686443104
entropy = 0.12263973142272337
brier score = 0.03404474790524376
adjusted brier score = 0.002073965841030571
entropy_class = 5.2984820104121875

```
Entropy for class 0: 0.007067635774381206
Entropy for class 1: 4.840260927205299
Entropy for class 2: 7.899645033293492
Entropy for class 3: 4.359957302826681
Entropy for class 4: 5.5370729832378185
Entropy for class 5: 4.083028349622112
Entropy for class 6: 10.88089807028977
Entropy for class 7: 4.7799257810479485
```

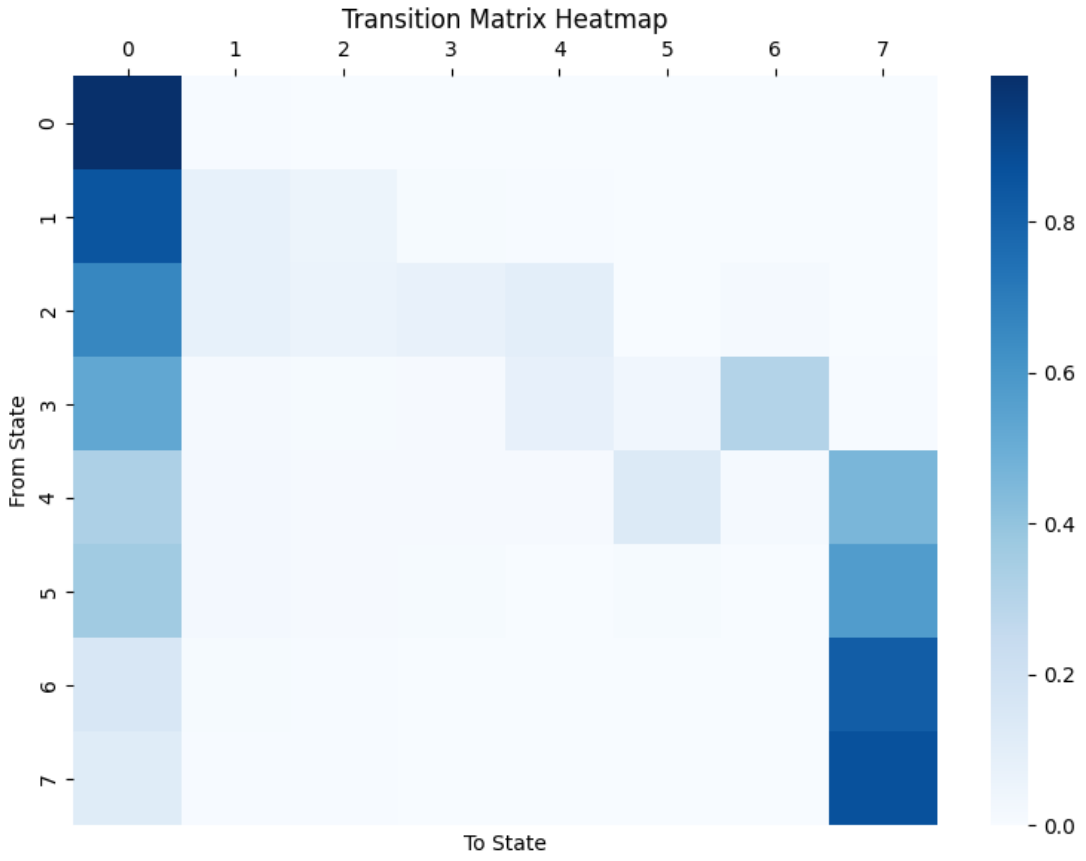
Transition Matrix Heat Map:



With 5 main features:
average probability = 0.253783938088625
entropy = 0.08870929325982928
brier score = 0.026583380972682508
adjusted brier score = 0.0014902885418081532
entropy_class = 3.988348267348335

```
Entropy for class 0: 0.010497314896987806
Entropy for class 1: 4.388829625150659
Entropy for class 2: 5.799355045062267
Entropy for class 3: 5.005564377349957
Entropy for class 4: 4.061381163542665
Entropy for class 5: 4.200182537971729
Entropy for class 6: 8.179186239120208
Entropy for class 7: 0.26178983569220704
```

Transition Matrix Heat Map:



2. encode `pred_proba` to one-hot

average probability = 0.24346980135274554
entropy = 0.09765431522867496
brier score = 0.031637665749323626
adjusted brier score = 0.00206502130955303
entropy_class = 4.008967180392805

```
Entropy for class 0: 0.009775633108989315
Entropy for class 1: 5.103320985277568
Entropy for class 2: 8.932882488210165
Entropy for class 3: 3.5038918716759784
Entropy for class 4: 3.420321908832572
Entropy for class 5: 5.703158799951571
Entropy for class 6: 5.242274037157176
Entropy for class 7: 0.1561117189284189
```

With `Credit Score`:
average probability = 0.2452789950399672
entropy = 0.10265724155852532 (1 months ahead: 0.05270378591909334)
brier score = 0.031817913828540344
adjusted brier score = 0.002108563723060312
entropy_class = 4.960439777564931

```
Entropy for class 0: 0.008615976046365109
Entropy for class 1: 5.078337751116721
Entropy for class 2: 8.70135446383711
Entropy for class 3: 4.153362048468214
Entropy for class 4: 3.394715197089275
Entropy for class 5: 3.7274792897850895
Entropy for class 6: 14.475913329089876
Entropy for class 7: 0.1437401650867989
```

With 5 main features:
average probability = 0.2501087228409682
entropy = 0.10682508970748208
brier score = 0.03168545357708819
adjusted brier score = 0.0020835252409971633
entropy_class = 4.9231551712164245

```
Entropy for class 0: 0.010108982486752996
Entropy for class 1: 4.965449773354866
Entropy for class 2: 10.365523092054913
Entropy for class 3: 4.588094649264331
Entropy for class 4: 3.8534709098834994
Entropy for class 5: 3.7038800713064925
Entropy for class 6: 11.778034990209454
Entropy for class 7: 0.12067890117108794
```

Iteration Method 2

***getting transition probabilities for all possible state transitions, then taking expected value based on previous transition probabilities*

```
def predict_n_months_weighted(mlp, n, input):
    features = input[['Credit Score']].to_numpy()

    # current distribution in one-hot form
    current_states = input[[column for column in input.columns if
column.startswith('Current Loan Delinquency Status')]].to_numpy()
    num_classes = current_states.shape[1]
    results = {}

    for month in range(n):
        # predict next month probability distributions
        input = np.hstack((current_states, features))
        next_states = mlp.predict_proba(input)

        # assume next state is 0,1...6+
        predicted_given_each_next_state = []
        for assumed_next_state in range(num_classes):
            one_hot_next_state = np.zeros_like(next_states)
            one_hot_next_state[:, assumed_next_state] = 1

            input_next = np.hstack((one_hot_next_state, features))
            predicted_probs = mlp.predict_proba(input_next)
            predicted_given_each_next_state.append(predicted_probs)
```

```
        # change list of arrays to 3d array
        predicted_given_each_next_state =
np.stack(predicted_given_each_next_state, axis=1)

        # weight by transition probabilities
        results = np.einsum('bi,bij->bj', next_states,
predicted_given_each_next_state)
        current_states = results

    return results

y_pred_proba = predict_n_months_weighted(mlp, MONTH_AHEAD, X_test)
```

average probability = 0.12526569605103838
entropy = 0.15076349984500884
brier score = 0.040581465430572984
adjusted brier score = 0.0034305335518346075
entropy_class = 6.761372954962091

```
Entropy for class 0: 0.009082499410059834
Entropy for class 1: 4.84262070067588
Entropy for class 2: 6.190071587924956
Entropy for class 3: 8.335823619830494
Entropy for class 4: 8.034283769444322
Entropy for class 5: 8.599714773707987
Entropy for class 6: 10.206425429927837
Entropy for class 7: 7.872961258775199
```

With `Credit Score`:
average probability = 0.12650117339075412
entropy = 0.1564331980069873
brier score = 0.040124938313709904
adjusted brier score = 0.0033586610557953143
entropy_class = 6.607209435350817

```
Entropy for class 0: 0.008355158580158721
Entropy for class 1: 4.762032437889109
Entropy for class 2: 5.822891885393652
Entropy for class 3: 6.348263259101781
Entropy for class 4: 6.774780175966556
Entropy for class 5: 9.07571327443016
Entropy for class 6: 9.803771246734502
Entropy for class 7: 10.261868044710617
```

With 5 main features:
average probability = 0.2036118311850642
entropy = 0.09982141095467027
brier score = 0.029232008014607735
adjusted brier score = 0.0017095765922018418
entropy_class = 4.6311793229218114

```
Entropy for class 0: 0.012030025664925564
Entropy for class 1: 4.564815739103751
Entropy for class 2: 5.403092226068231
Entropy for class 3: 5.422463509341074
Entropy for class 4: 6.221682428220605
Entropy for class 5: 6.7556407893178925
Entropy for class 6: 7.768415110114107
Entropy for class 7: 0.9012947555439064
```

**Train MLP with status 3 months ago

We can also modifiy the training process according the months to predict ahead, as:

```
MONTH_AHEAD = 3
# in function preprocess()
df['Next Loan Delinquency Status'] = df.groupby('Loan Sequence Number')['Current
Loan Delinquency Status'].shift(-MONTH_AHEAD)
```

average probability = 0.30554111135043394
entropy = 0.0793874576832437
brier score = 0.02810046936215188
adjusted brier score = 0.001936038324794717
entropy_class = 2.6362843605049604

With `Credit Score`:
average probability = 0.31309216414193974
entropy = 0.07540912794051383
brier score = 0.02790038888099853
adjusted brier score = 0.0019538893776107928
entropy_class = 2.5246880254080595

With 5 main features:
average probability = 0.3154518015108996
entropy = 0.07496824565015968
brier score = 0.027976407199123924
adjusted brier score = 0.0019539852498494854
entropy_class = 2.4883358783761147