```python
import pandas as pd
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.metrics import log_loss

data = pd.read_csv("Test_Data COVID.csv")
pd.set_option('display.max_columns', None)

data = data.iloc[:, :28]
data.info()
data.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 509 entries, 0 to 508
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Test_Positive                   509 non-null    int64
 1   Gender                          509 non-null    int64
 2   Age                             509 non-null    int64
 3   swelling                        509 non-null    int64
 4   loss_of_appetite                509 non-null    int64
 5   Chest_pain                      509 non-null    int64
 6   Chills                          509 non-null    int64
 7   Cough                           509 non-null    int64
 8   Diarrhea                        509 non-null    int64
 9   Difficulty_breathing            509 non-null    int64
 10  Excessive_sweating              509 non-null    int64
 11  Fatigue                         509 non-null    int64
 12  Fever                           509 non-null    int64
 13  Headaches                       509 non-null    int64
 14  Joint_pain                      509 non-null    int64
 15  Loss_of_balance                 509 non-null    int64
 16  Loss_of_taste                   509 non-null    int64
 17  Loss_of_smell                   509 non-null    int64
 18  Muscle_aches                    509 non-null    int64
 19  New_confusion                   509 non-null    int64
 20  Pinkeye_or_Conjunctivitis       509 non-null    int64
 21  Runny_nose                      509 non-null    int64
 22  Shortness_of_breath             509 non-null    int64
 23  Sore_throat                     509 non-null    int64
 24  Stomach_or_abdominal_pain       509 non-null    int64
 25  Unusual_shivering_or_shaking    509 non-null    int64
 26  Nausea_or_vomiting              509 non-null    int64
 27  Wheezing                        509 non-null    int64
dtypes: int64(28)
memory usage: 111.5 KB

(509, 28)
```

```
print("Answer b.\n")

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting',
           'Joint_pain', 'loss_of_appetite', 'Wheezing',
           'Difficulty_breathing', 'Excessive_sweating']

X = data [columns]
y = data ['Test_Positive']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Answer b.

Fever: 0.105
Chills: 0.116
Muscle_aches: 0.085
Cough: 0.174
loss_of_appetite: 0.142
Difficulty_breathing: 0.061
Excessive_sweating: 0.054

McFadden's Pseudo R-squared:  0.341

Intercept:  0.028
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting',
           'Joint_pain', 'loss_of_appetite', 'Wheezing',
           'Difficulty_breathing']

X = data [columns]
y = data ['Excessive_sweating']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.109
Fever: 0.146
Joint_pain: 0.192
Difficulty_breathing: 0.241

McFadden's Pseudo R-squared:  0.641

Intercept:  0.006
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting',
           'Joint_pain', 'loss_of_appetite', 'Wheezing']
```

```
X = data [columns]
y = data ['Difficulty_breathing']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Cough: 0.108
Shortness_of_breath: 0.265
loss_of_appetite: 0.236

McFadden's Pseudo R-squared:  0.571

Intercept:  0.006
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting',
           'Joint_pain', 'loss_of_appetite']

X = data [columns]
y = data ['Wheezing']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)
```

```
coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Loss_of_taste: 0.065
Shortness_of_breath: 0.14

McFadden's Pseudo R-squared:  0.54

Intercept:  0.005
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting',
           'Joint_pain']

X = data [columns]
y = data ['loss_of_appetite']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)
```

```
null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Loss_of_taste: 0.257
Loss_of_smell: 0.11
Muscle_aches: 0.055
Diarrhea: 0.098
Cough: 0.099
Shortness_of_breath: 0.126
Loss_of_balance: 0.207

McFadden's Pseudo R-squared:  0.677

Intercept:  0.005
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance',
'Nausea_or_vomiting']

X = data [columns]
y = data ['Joint_pain']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)
```

```
print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Chills: 0.084
Muscle_aches: 0.205
Loss_of_balance: 0.069

McFadden's Pseudo R-squared:  0.608

Intercept:  0.002
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat', 'Loss_of_balance']

X = data [columns]
y = data ['Nausea_or_vomiting']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Muscle_aches: 0.14
Diarrhea: 0.171

McFadden's Pseudo R-squared:  0.504

Intercept:  0.005
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose', 'Sore_throat']

X = data [columns]
y = data ['Loss_of_balance']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Loss_of_smell: 0.077
Shortness_of_breath: 0.194

McFadden's Pseudo R-squared:  0.517

Intercept:  0.005
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath',
           'Runny_nose']

X = data [columns]
y = data ['Sore_throat']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)
```

```
y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.121
Headaches: 0.15
Runny_nose: 0.374

McFadden's Pseudo R-squared:  0.614

Intercept:  0.009
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough', 'Shortness_of_breath']

X = data [columns]
y = data ['Runny_nose']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)
```

```
null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.293
Headaches: 0.11
Cough: 0.257

McFadden's Pseudo R-squared:  0.57

Intercept:  0.013
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea', 'Cough']

X = data [columns]
y = data ['Shortness_of_breath']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.115
Loss_of_smell: 0.056
Muscle_aches: 0.154
Cough: 0.064
```

```
McFadden's Pseudo R-squared:  0.548

Intercept:  -0.001

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches', 'Diarrhea']

X = data [columns]
y = data ['Cough']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))

Fatigue: 0.242
Headaches: 0.339
Chills: 0.256

McFadden's Pseudo R-squared:  0.441

Intercept:  0.048

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills',
           'Muscle_aches']

X = data [columns]
y = data ['Diarrhea']
```

```
lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.149
Chills: 0.165
Muscle_aches: 0.09

McFadden's Pseudo R-squared:  0.403

Intercept:  0.009
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell',
'Chills']

X = data [columns]
y = data ['Muscle_aches']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")
```

```
log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.254
Headaches: 0.071
Chills: 0.323

McFadden's Pseudo R-squared:  0.627

Intercept:  0.005
```

```
print("Answer g.\n")

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches', 'Loss_of_smell']

X = data [columns]
y = data ['Chills']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Answer g.

Fatigue: 0.349
Fever: 0.365

McFadden's Pseudo R-squared:  0.57

Intercept:  0.015

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever', 'Headaches']

X = data [columns]
y = data ['Loss_of_smell']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))

Loss_of_taste: 0.659
Fever: 0.092

McFadden's Pseudo R-squared:  0.648

Intercept:  0.009

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste', 'Fever']

X = data [columns]
y = data ['Headaches']
```

```
lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.591
Fever: 0.11

McFadden's Pseudo R-squared:  0.485

Intercept:  0.042
```

```
print("Answer e.\n")

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue',
           'Loss_of_taste']

X = data [columns]
y = data ['Fever']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")
```

```
log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Answer e.

Fatigue: 0.316
Loss_of_taste: 0.055

McFadden's Pseudo R-squared:  0.393

Intercept:  0.022
```

```
columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking', 'Fatigue']

X = data [columns]
y = data ['Loss_of_taste']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Fatigue: 0.223

McFadden's Pseudo R-squared:  0.496
```

```
Intercept:  0.001

columns = ['Gender', 'Age', 'Unusual_shivering_or_shaking']

X = data [columns]
y = data ['Fatigue']

lasso = Lasso(alpha = 0.0005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Unusual_shivering_or_shaking: 0.812

McFadden's Pseudo R-squared:  0.051

Intercept:  0.118
```

```
columns = ['Gender', 'Age']

X = data [columns]
y = data ['Unusual_shivering_or_shaking']

lasso = Lasso(alpha = 0.005)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
```

```
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
McFadden's Pseudo R-squared:  0.0

Intercept:  0.006
```

```
print("Answer h.\n")

columns = ['Fatigue']

X = data [columns]
y = data ['Chills']

lasso = Lasso(alpha = 0.0056)
lasso.fit(X, y)

y_pred = lasso.predict(X)

coefficients = lasso.coef_
column_names = X.columns

for feature, coef in zip(column_names, coefficients):
  rounded_coef = round(coef, 3)
  if abs(rounded_coef) > 0.05:
    print(f"{feature}: {rounded_coef}")

log_likelihood_model = -log_loss(y, y_pred)

null_model_probs = np.full_like(y_pred, y.mean())
log_likelihood_null_model = -log_loss(y, null_model_probs)

mcfadden_r2 = 1 - (log_likelihood_model / log_likelihood_null_model)

print('\nMcFadden\'s Pseudo R-squared: ', mcfadden_r2.round(3))
print('\nIntercept: ', lasso.intercept_.round(3))
```

```
Answer h.
```

```
Fatigue: 0.484

McFadden's Pseudo R-squared:  0.462

Intercept:  0.019
```

```
print("Answer b.\nFrequency for Fatigue and Loss of Taste:\n")

Fatigue_counts = data['Fatigue'].value_counts()
print(Fatigue_counts)
Fatigue_frequency = round(50/(459+50), 3)
print("Frequency for Fatigue: ", Fatigue_frequency, "\n")

LossOfTaste_counts = data['Loss_of_taste'].value_counts()
print(LossOfTaste_counts)
LossOfTaste_frequency = round(14/(495+14), 3)
print("Frequency for Loss of Taste: ", LossOfTaste_frequency)
```

```
Answer b.
Frequency for Fatigue and Loss of Taste:

0    459
1     50
Name: Fatigue, dtype: int64
Frequency for Fatigue:  0.098

0    495
1     14
Name: Loss_of_taste, dtype: int64
Frequency for Loss of Taste:  0.028
```