Logistic Regression

In this notebook you will use GPU-accelerated logistic regression to predict infection risk based on features of our population members.

Objectives

By the time you complete this notebook you will be able to:

- Use GPU-accelerated logistic regression

Imports

In [1]:
```
import cudf
import cuml

import cupy as cp
```

Load Data

In [2]:
```
gdf = cudf.read_csv('./data/pop_2-05.csv', usecols=['age', 'sex', 'infected'])
```

In [3]:
```
gdf.dtypes
```

Out[3]:
```
age         float64
sex         float64
infected    float64
dtype: object
```

In [4]:
```
gdf.shape
```

Out[4]:
```
(58479894, 3)
```

In [5]:
```
gdf.head()
```

Out[5]:

	age	sex	infected
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0

Logistic Regression

Logistic regression can be used to estimate the probability of an outcome as a function of some (assumed independent) inputs. In our case, we would like to estimate infection risk based on population members' age and sex.

Here we create a cuML logistic regression instance `logreg` :

In [6]:
```
logreg = cuml.LogisticRegression()
```

Exercise: Regress Infected Status

The `logreg.fit` method takes 2 arguments: the model's independent variables *X*, and the dependent variable *y*. Fit the `logreg` model using the `gdf` columns `age` and `sex` as *X* and the `infected` column as *y*.

In [8]:
```
logreg.fit(gdf[['age', 'sex']], gdf['infected'])
```

Out[8]:
```
LogisticRegression()
```

Solution

In [9]:
```
# %load solutions/regress_infected
logreg.fit(gdf[['age', 'sex']], gdf['infected'])
```

Out[9]:
```
LogisticRegression()
```

Viewing the Regression

After fitting the model, we could use `logreg.predict` to estimate whether someone has more than a 50% chance to be infected, but since the virus has low prevalence in the population (around 1-2%, in this data set), individual probabilities of infection are well below 50% and the model should correctly predict that no one is individually likely to have the infection.

However, we also have access to the model coefficients at `logreg.coef_` as well as the intercept at `logreg.intercept_` . Both of these values are cuDF Series:

In [10]:
```
type(logreg.coef_)
```

Out[10]:
```
cudf.core.series.Series
```

In [11]:
```
type(logreg.intercept_)
```

Out[11]:
```
cudf.core.series.Series
```

Here we view these values. Notice that changing sex from 0 to 1 has the same effect via the coefficients as changing the age by ~48 years.

In [12]:

```
logreg_coef = logreg.coef_
logreg_int = logreg.intercept_

print("Coefficients: [age, sex]")
print([logreg_coef[0], logreg_coef[1]])

print("Intercept:")
print(logreg_int[0])
```

```
Coefficients: [age, sex]
[0.014860597365798499, 0.6956658839479832]
Intercept:
-5.222369426097303
```

Estimate Probability of Infection

As with all logistic regressions, the coefficients allow us to calculate the logit for each; from that, we can calculate the estimated percentage risk of infection.

In [13]:

```
class_probs = logreg.predict_proba(gdf[['age', 'sex']])
class_probs
```

Out[13]:

	0	1
0	0.994634	0.005366
1	0.994634	0.005366
2	0.994634	0.005366
3	0.994634	0.005366
4	0.994634	0.005366
...	...	...
58479889	0.960428	0.039572
58479890	0.960428	0.039572
58479891	0.960428	0.039572
58479892	0.960428	0.039572
58479893	0.960428	0.039572

58479894 rows × 2 columns

Remembering that a 1 indicates 'infected', we assign that class' probability to a new column in the original dataframe:

In [14]:

```
gdf['risk'] = class_probs[1]
```

Looking at the original records with their new estimated risks, we can see how estimated risk varies across individuals.

In [15]:
```
gdf.take(cp.random.choice(gdf.shape[0], size=5, replace=False))
```

Out[15]:

	age	sex	infected	risk
57146212	82.0	1.0	0.0	0.035293
27742448	80.0	0.0	1.0	0.017404
3885527	10.0	0.0	0.0	0.006220
11693027	31.0	0.0	0.0	0.008479
8476466	23.0	0.0	0.0	0.007535

Exercise: Show Infection Prevalence is Related to Age

The positive coefficient on age suggests that the virus is more prevalent in older people, even when controlling for sex.

For this exercise, show that infection prevalence has some relationship to age by printing the mean `infected` values for the oldest and youngest members of the population when grouped by age:

In [17]:
```
age_groups = gdf[['age', 'infected']].groupby(['age'])
print(age_groups.mean().head())
print(age_groups.mean().tail())
```

```
      infected
age
66.0  0.020700
71.0  0.021292
82.0  0.022929
64.0  0.020675
77.0  0.022102
      infected
age
33.0  0.015707
76.0  0.021928
74.0  0.021807
79.0  0.022518
86.0  0.023417
```

Solution

In [18]:
```
# %load solutions/risk_by_age
age_groups = gdf[['age', 'infected']].groupby(['age'])
print(age_groups.mean().head())
print(age_groups.mean().tail())
```

```
      infected
age
66.0  0.020700
71.0  0.021292
82.0  0.022929
64.0  0.020675
77.0  0.022102
      infected
age
33.0  0.015707
76.0  0.021928
74.0  0.021807
79.0  0.022518
86.0  0.023417
```

Exercise: Show Infection Prevalence is Related to Sex

Similarly, the positive coefficient on sex suggests that the virus is more prevalent in people with sex = 1 (females), even when controlling for age.

For this exercise, show that infection prevalence has some relationship to sex by printing the mean `infected` values for the population when grouped by sex:

In [20]:
```
sex_groups = gdf[['sex', 'infected']].groupby(['sex'])
sex_groups.mean()
```

Out[20]:

sex	infected
0.0	0.010140
1.0	0.020713

Solution

In [21]:
```
# %load solutions/risk_by_sex
sex_groups = gdf[['sex', 'infected']].groupby(['sex'])
sex_groups.mean()
```

Out[21]:

sex	infected
0.0	0.010140
1.0	0.020713

Making Predictions with Separate Training and Test Data

cuML gives us a simple method for producing paired training/testing data:

In [22]:
```
X_train, X_test, y_train, y_test  = cuml.train_test_split(gdf[['age', 'sex']], gdf['ir
```

Exercise: Fit Logistic Regression Model Using Training Data

For this exercise, create a new logistic regression model `logreg` , and fit it with the *X* and *y* training data just created.

In [24]:
```
logreg = cuml.LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[24]:
```
LogisticRegression()
```

Solution

In [25]:
```
# %load solutions/fit_training
logreg = cuml.LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[25]:
```
LogisticRegression()
```

Use Test Data to Validate Model

We can now use the same procedure as above to predict infection risk using the test data:

In [26]:
```
y_test_pred = logreg.predict_proba(X_test, convert_dtype=True)[1]
y_test_pred.index = X_test.index
y_test_pred
```

Out[26]:
```
2456343     0.005864
4435961     0.006408
44361586    0.020084
32379232    0.012217
30563426    0.011353
              ...
22886826    0.013178
11427342    0.008480
10943095    0.008233
38600339    0.016137
32198045    0.012217
Name: 1, Length: 5847990, dtype: float64
```

As we saw before, very few people are actually infected in the population, even among the highest-risk groups. As a simple way to check our model, we split the test set into above-average predicted risk and below-average predicted risk, then observe that the prevalence of infections correlates closely to those predicted risks.

In [27]:
```
test_results = cudf.DataFrame()
test_results['age'] = X_test['age']
test_results['sex'] = X_test['sex']
test_results['infected'] = y_test
```

```
test_results['predicted_risk'] = y_test_pred

test_results['high_risk'] = test_results['predicted_risk'] > test_results['predicted_r

risk_groups = test_results.groupby('high_risk')
risk_groups.mean()
```

Out[27]:

	age	sex	infected	predicted_risk
high_risk				
True	56.173038	0.890247	0.023814	0.023315
False	29.520700	0.252492	0.009972	0.010325

Finally, in a few milliseconds, we can do a two-tier analysis by sex and age:

In [28]:
```
%%time
s_groups = test_results[['sex', 'age', 'infected', 'predicted_risk']].groupby(['sex',
s_groups.mean()
```

```
CPU times: user 9.89 ms, sys: 8.34 ms, total: 18.2 ms
Wall time: 17.2 ms
```

Out[28]:

		infected	predicted_risk
sex	**age**		
0.0	**17.0**	0.007454	0.006898
	13.0	0.005125	0.006503
1.0	**18.0**	0.014355	0.013940
0.0	**39.0**	0.012146	0.009539
1.0	**42.0**	0.023505	0.019794
	...	...	...
	50.0	0.023546	0.022236
0.0	**27.0**	0.009161	0.007994
1.0	**77.0**	0.029074	0.032848
	54.0	0.025037	0.023565
0.0	**29.0**	0.010696	0.008233

182 rows × 2 columns

Please Restart the Kernel

In [29]:
```
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

Out[29]:
```
{'status': 'ok', 'restart': True}
```

Next

In the next notebook, you will use GPU-accelerated k-nearest-neighbors algorithm to locate the nearest road nodes to each hospital.