Jupyter: Intro to Data Science - Lecture 6/7 Logistic Regression

Grant Long
CUNY City College

NYC Tech-in-Residence Corps

Follow this and additional works at: https://academicworks.cuny.edu/cc_oers

How does access to this work benefit you? Let us know!

Recommended Citation
https://academicworks.cuny.edu/cc_oers/162

This Lecture or Presentation is brought to you for free and open access by the City College of New York at CUNY Academic Works. It has been accepted for inclusion in Open Educational Resources by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@cuny.edu.
Data Dive Week 6/7: Logistic Regression

This week we take a look at logistic regression, the first classification model we'll be covering in class. We'll be using scikit-learn in today's exercise.

As we discussed last week, logistic regression is a classification model - meaning that it is designed to identify the likelihood that a given observed data point belongs to set class, or category. Today we'll be looking at a real world application of logistic regression using data from loans requests posted on Kiva.org (kiva.org).

Kiva is an international nonprofit, founded in 2005 and based in San Francisco, with a mission to connect people through lending to alleviate poverty. Kiva seeks to celebrate and support people looking to create a better future for themselves, their families and their communities.

By lending as little as $25 on Kiva, anyone can help a borrower start or grow a business, go to school, access clean energy or realize their potential. For some, it's a matter of survival, for others it's the fuel for a life-long ambition.

Today's Modeling Objective

Our focus today will be determining whether microfinance projects on the site Kiva.org (kiva.org) receive funding or not using a host of features made available by Kiva, along with some features we'll design ourselves.

Today's data is a subsample of projects in Kenya, one of Kiva's most active countries for lending. We'll be working with 18,000 observation, 3,000 or which were funded.

Documentation on the data is available on Kiva's website (http://build.kiva.org/docs/data/basic_types).
Tidying Up Our Data

There are a couple of different things we'll want to off the bat:

1. Create a target variable
2. Generate a more usable version of the POSTED_TIME column.
3. Generate a variable with the amount of planned time before expiration for each project.

Based on a quick look at the data, what else might be interesting to try off the bat?

Data Exploration

Which columns might be helpful in predicting successful funding? How do we know?
Training A Model in `statsmodels`

As we saw the past two weeks, `statsmodels` can be helpful if we want to visualize the summary statistics of our output.

First, let's identify the columns we're interested in and extract the rows with valid results. We'll also want to add a constant column.

```python
var = 'ACTIVITY_NAME'
df[~df['success'].isna() & df['success']==1]
df[['var', 'success']].groupby('var')
  .agg([('count', 'mean')])
  .sort_values(by=('success', 'count'), ascending=False)
  .head(10)
```

Next, let's use `statsmodels` to fit the model and print the result.

```python
logit = sm.Logit(df.loc[valids, 'success'], df.loc[valids, model_columns])
result = logit.fit()
print(result.summary())
```
Training A Model in sklearn

As we saw the past two weeks, statsmodels can be helpful if we want to visualize the summary statistics of our output. Documentation for the LogisticRegression object can be found here (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html).

Let's fit and score the model:

```
In [ ]:
model_columns = ['LOAN_AMOUNT', 'posted_year', 'posted_duration', 'LENDER_T
valids = df[model_columns].notna().all(axis=1)
X = df.loc[valids, model_columns].values
y = df.loc[valids, 'success'].values

clf = LogisticRegression(random_state=20181022).fit(X, y)
print('The accuracy of our model is %0.1f%%' % (clf.score(X, y)*100))
```

Let's see where we went right and where we went wrong:

```
In [ ]:
df.loc[valids, 'likelihood'] = clf.predict_proba(X)[:,:1]

Let's take a look at the top values where we predicted incorrectly predicted a failure to fund:

```
In [ ]:
df.loc[(valids & (df.success==1)), ].sort_values(by='likelihood').head()
```

Now let's take a look at where we predicted funding, but the projects were not funded.

```
In [ ]:
df.loc[(valids & (df.success==0)), ].sort_values(by='likelihood').tail()
```

Finally, let's take a closer look at our summary metrics. How do they change as we change our cutoff value for our prediction?
def calculate_metrics(df, threshold):
    df['predicted'] = (clf.predict_proba(X)[:,1]>=threshold)*1
    accuracy = sum(df['predicted']==y)/len(y)
    precision = df.loc[df.predicted==1, 'success'].mean()
    recall = df.loc[df.success==1, 'predicted'].mean()

    return accuracy, precision, recall, sum(df['predicted'])

for p in [.4, .5, .6, .9]:
    print(p, calculate_metrics(df.loc[valids,].copy(), p))