7_Improving_classification_models

October 5, 2021

1 Exercise: Improving a logistic regression model

In the previous exercise, we fit a simple logistic regression model to predict the chance of an avalanche. This time, we'll improve its performance by using multiple features intelligently.

1.1 Data visualisation

Let's load our data.

```
[ ]: import pandas
     import graphings

     #Import the data from the .csv file
     dataset = pandas.read_csv('avalanche.csv', delimiter="\t", index_col=0)

     # Split our data into training and test
     import sklearn.model_selection
     train, test = sklearn.model_selection.train_test_split(dataset, test_size=0.25,␣
      ↪random_state=10)

     print("Train size:", train.shap[0])
     print("Test size:", test.shape[0])

     #Let's have a look at the data
     print(train.head())
```

```
     ␣
↪---------------------------------------------------------------------------

    AttributeError                            Traceback (most recent call␣
↪last)

    <ipython-input-1-96a90fa61285> in <module>
      9 train, test = sklearn.model_selection.train_test_split(dataset,␣
↪test_size=0.25, random_state=10)
     10
---> 11 print("Train size:", train.shap[0])
     12 print("Test size:", test.shape[0])
```

```
     13


    ~\anaconda3\lib\site-packages\pandas\core\generic.py in␣
↪__getattr__(self, name)
   5463             if self._info_axis.
↪_can_hold_identifiers_and_holds_name(name):
   5464                 return self[name]
-> 5465             return object.__getattribute__(self, name)
   5466
   5467     def __setattr__(self, name: str, value) -> None:


AttributeError: 'DataFrame' object has no attribute 'shap'
```

We have numerous features available:

- `surface_hoar` is how disturbed the surface of the snow is
- `fresh_thickness` is how thick the top layer of snow is, or 0 if there's no fresh snow on top
- `wind` is the top wind speed that day, in km/h
- `weak_layers` is the number of layers of snow that aren't well-bound to other layers
- `no_visitors` is the number of hikers who were on the trail that day
- `tracked_out` is a `1` or `0`. A `1` means that the snow has been trampled heavily by hikers

1.2 Simple logistic regression

Let's make a simple logistic regression model and assess its performance with accuracy.

[ ]:
```python
import sklearn
from sklearn.metrics import accuracy_score
import statsmodels.formula.api as smf

# Perform logistic regression.
model = smf.logit("avalanche ~weak_layers",train).fit()

# Calculate accuracy
def calculate_accuracy(model):
    '''
    Calculates accuracy
    '''

    # Make estimations and convert to categories
    avalanche_predicted = model.predict(test)> 0.5

    # Calculate what proportion were predicted correctly
    # We can use skearn to calculate accuracy for us
    print("Accuracy:", accuracy_score(test.avalanche,avalanche_predicted))
```

```
calculate_accuracy(model)
```

```
Optimization terminated successfully.
         Current function value: 0.616312
         Iterations 5
Accuracy: 0.6167883211678832
```

Let's see how we can improve our model

1.3 Utilizing multiple features

Most of our features seem like they could be useful, least in theory. Let's try a model with all features we've available.

```
[ ]: # Perform logistic regression.
model_all_features= smf.logit("avalanche ~ weak_layers + fresh_thickness + wind␣
 ↪+ surface_hoar + no_visitors + tracked_out",train).fit()
calculate_accuracy(model_all_features)
```

```
Optimization terminated successfully.
         Current function value: 0.459347
         Iterations 7
Accuracy: 0.7846715328467153
```

That's a big improvement on the simpler model we've been working with.

To understand why, we can look at the summary information

```
[ ]: model_all_features.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              avalanche   No. Observations:                  821
Model:                          Logit   Df Residuals:                      814
Method:                           MLE   Df Model:                            6
Date:                Tue, 05 Oct 2021   Pseudo R-squ.:                  0.3305
Time:                        10:59:23   Log-Likelihood:                -377.12
converged:                       True   LL-Null:                       -563.33
Covariance Type:            nonrobust   LLR p-value:                 2.372e-77
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept          -4.0107      0.443     -9.043      0.000      -4.880
-3.141
weak_layers         0.3733      0.034     10.871      0.000       0.306
```

```
0.441
fresh_thickness    -0.0220      0.030     -0.732      0.464      -0.081
0.037
wind                0.1009      0.009     11.149      0.000       0.083
0.119
surface_hoar        0.3306      0.035      9.424      0.000       0.262
0.399
no_visitors        -0.1060      0.032     -3.262      0.001      -0.170
-0.042
tracked_out        -0.0664      0.181     -0.367      0.713      -0.420
0.288
================================================================================
===
"""
```

Take a look at the P column, recalling that values less than 0.05 mean we can be confident that this parameter is helping the model make better predictions.

Both `surface_hoar` and `wind` have very small values here, meaning they're useful predictors and probably explain why our model is working better. If we look at the `coef` (which states *parameters*) column we see that these have positive values. This means that higher winds, and greater amounts of surface hoar result in higher avalanche risk.

1.4 Simplifying our model

Looking at the summary again, we can see that `tracked_out` (how trampled the snow is), and `fresh_thickness` have large p-values. This means they aren't useful predictors. Let's see what happens if we remove them from our model:

[ ]:
```
# Perform logistic regression.
model_simplified = smf.logit("avalanche ~ weak_layers + surface_hoar + wind +␣
↪no_visitors", train).fit()
calculate_accuracy(model_simplified)
```

```
Optimization terminated successfully.
         Current function value: 0.459760
         Iterations 7
Accuracy: 0.781021897810219
```

Our new model works very similarly to the old one! In some circumstances simplifying a model like this can even improve it, as it becomes less likely to overfit.

1.5 Careful feature selection

Usually, we don't just pick features blindly. Let's think about what we've just done - we removed how much fresh snow was in a model, trying to predict avalanches. Something seems off. Surely avalanches are much *more* likely after it has snowed? Similarly, the number of people on the track seems unrelated to how many avalanches there were, but we know that people often can trigger avalanches.

Let's review our earlier model again:

```
[ ]: model_all_features.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              avalanche   No. Observations:                  821
Model:                          Logit   Df Residuals:                      814
Method:                           MLE   Df Model:                            6
Date:                Tue, 05 Oct 2021   Pseudo R-squ.:                  0.3305
Time:                        11:04:19   Log-Likelihood:                -377.12
converged:                       True   LL-Null:                       -563.33
Covariance Type:            nonrobust   LLR p-value:                 2.372e-77
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept          -4.0107      0.443     -9.043      0.000      -4.880
-3.141
weak_layers         0.3733      0.034     10.871      0.000       0.306
0.441
fresh_thickness    -0.0220      0.030     -0.732      0.464      -0.081
0.037
wind                0.1009      0.009     11.149      0.000       0.083
0.119
surface_hoar        0.3306      0.035      9.424      0.000       0.262
0.399
no_visitors        -0.1060      0.032     -3.262      0.001      -0.170
-0.042
tracked_out        -0.0664      0.181     -0.367      0.713      -0.420
0.288
================================================================================
===
"""
```

Look at the `fresh_thickness` row. We're told that it has a negative coefficient. This means that as thickness increases, avalanches decrease.

Similarly, `no_visitors` has a negative coefficient, meaning that fewer hikers means more avalanches.

How can this bes? Well, while visitors can cause avalanches if there's a lot of fresh snow, presumably they cannot do so easily if there's no fresh snow. This means that our features aren't fully independent.

We can tell the model to try to take into account that these features interact, using a multiply sign. Let's try that now.

```
[ ]: # Create a model with an interaction. Notice the end of the string where
     # we've a multiply sign between no_visitors and fresh_thickness
     formula = "avalanche ~ weak_layers + surface_hoar + wind + no_visitors *␣
      ↪fresh_thickness"
     model_with_interaction = smf.logit(formula, train).fit()
     calculate_accuracy(model_with_interaction)
```

```
Optimization terminated successfully.
         Current function value: 0.413538
         Iterations 7
Accuracy: 0.8357664233576643
```

The model has improved to 84% accuracy! Let's look at the summary information:

```
[ ]: model_with_interaction.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
     """
                                Logit Regression Results
     ==============================================================================
     Dep. Variable:              avalanche   No. Observations:                  821
     Model:                          Logit   Df Residuals:                      814
     Method:                           MLE   Df Model:                            6
     Date:                Tue, 05 Oct 2021   Pseudo R-squ.:                  0.3973
     Time:                        11:08:37   Log-Likelihood:                -339.51
     converged:                       True   LL-Null:                       -563.33
     Covariance Type:            nonrobust   LLR p-value:                 1.587e-93
     ================================================================================
     ===============
                                      coef    std err          z      P>|z|
     [0.025      0.975]
     --------------------------------------------------------------------------------
     ---------------
     Intercept                     -0.9606      0.587     -1.636      0.102
     -2.111       0.190
     weak_layers                    0.4327      0.039     11.193      0.000
     0.357       0.508
     surface_hoar                   0.3887      0.039     10.035      0.000
     0.313       0.465
     wind                           0.1204      0.010     11.607      0.000
     0.100       0.141
     no_visitors                   -0.9430      0.114     -8.237      0.000
     -1.167      -0.719
     fresh_thickness               -0.4962      0.069     -7.191      0.000
     -0.631      -0.361
     no_visitors:fresh_thickness    0.1015      0.013      7.835      0.000
     0.076       0.127
     ================================================================================
```

```
===============
"""
```

We can see that the interaction term is helpful - the p-value is less than 0.05. The model is also performing better than our previous attempts.

1.6 Making predictions with multiple features

Very quickly, lets explore what this interaction means by looking at model predictions.

We will first graph two independent features in 3D. Let's start with `weak_layers` and `wind`:

[ ]:
```
graphings.model_to_surface_plot(model_with_interaction, ["weak_layers",␣
↪"wind"], test)
```

```
Creating plot…
```

The graph is interactive - rotate it and explore how there's a clear s-shaped relationship between the features and probability.

Let's now look at the features that we've said can interact:

[ ]:
```
graphings.model_to_surface_plot(model_with_interaction, ["no_visitors",␣
↪"fresh_thickness"], test)
```

```
Creating plot…
```

It looks quite different to the other! From any side, we can see an s-shape, but these combine in strange ways.

We can see that the risk goes up on days with lots of visitors *and* lots of snow. There is no real risk of avalanche when there's a lot of snow but no visitors, or when there are a lot of visitors but no snow.

The fact that it shows high risk when there's no fresh snow and no visitors could be due to rain, which keeps visitors and snow clouds away but results in avalanches of the older snow. To confirm this, we'd need to explore the data in more depth, but we'll stop here for now.

1.7 Summary

Well done! Let's recap. We've:

- improved our simple model by adding more features.
- practiced interpreting our model coefficients (parameters) from the model summary
- eliminated unnecessary features
- explored how sometimes it's important to think about what your data really mean
- created a model that combined features to give superior result