Lending Club Machine Learning Model and API

23 August 2017 / Projects / Machine Learning

This is an API that provides a probability of default when presented a loan from Lending Club. The API is written in Flask and it utilizes a scikit-learn machine learning model.

You can get the code for this at the GitHub repo (https://github.com/orangganjil/lendingclub-default-predictor).

Prerequisites

- Python 3
- Flask
- Flask-RESTful
- numpy
- scipy
- pandas
- scikit-learn (0.19.0) - pickled model may be version dependent

Installation

Installation is completely up to you but it expects the typical Flask environment with the two pickled scikit-learn models in the same directory as the Flask app (this can be changed by modifying the joblib path within the app).

Usage

The API expects a JSON file conforming to the current format from Lending Club. There is an example JSON file called `loanlist-example.json` showing the format the model expects.

There are three endpoints to the API:

- / (provides a readme/description)
- /version (provides a JSON output of the API version)
- /predict (this is the predictor)

To use the API, POST a JSON file matching the format of the example file provided. Be sure to set your Content-Type to `application\json` for the POST.

The API will then load the loans into a Pandas DataFrame, make some modifications, and then run them through the Random Forest Classifier model. It then will add another column to the DataFrame for the predicted probability of default/charge-off. Finally, the API will then return JSON output of the memberId and probability of default for each loan that was input.

Machine Learning Model

The machine learning model is a Random Forest Classifier trained on all Lending Club loans from 2007 through the first half of 2017. It has the following parameters:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_split=1e-07,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=3,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Below is the confusion matrix followed by the accuracy, AUC, precision, and recall scores for the model:

True Positive False Positive	False Negative True Negative
74,319	4,546
1,870	16,393

```
Accuracy: 0.933942838316
Error: 0.0660571616836
AUC: 0.919982188297
Precision: 0.782893165863
Recall: 0.897607183924
```

If you are unfamiliar with a confusion matrix, in a binary decision such as whether an account is likely to default, the matrix lists in the first column the number of true positives over false positives. The second column lists the number of false negatives over true negatives. Ideally, one would like the numbers to be greatest going diagonally from top-left to bottom-right, with zeros in the other two cells.

You can read more about a confusion matrix on Wikipedia (https://en.wikipedia.org/wiki/Confusion_matrix).

The two models used in the API are the following:

Label Encoder

Used for both the "term" and "grade" columns/features to encode them into labels. This, rather than one-hot encoding, was used due to the simplicity of the features and in an attempt to keep the feature size to a minimum (one of my goals was to enable this to run on a Raspberry Pi). It is important that this label encoder be loaded so that labels for new loans are consistent with those of the training set.

Random Forest Classifier

This model is a random forest classifier with 100 trees. It does the predicting, returning an array consisting of the probability a loan will not default along with the probability it will. The API is only returning the probability of default - the second field returned by the model.

Performance

The classifier is pretty quick and should work just fine on even a Raspberry Pi.

Have a question or suggestion?

Send me a message directly on Twitter (https://twitter.com/codyhatch).