Homework 1

628 Machine Learning in Finance

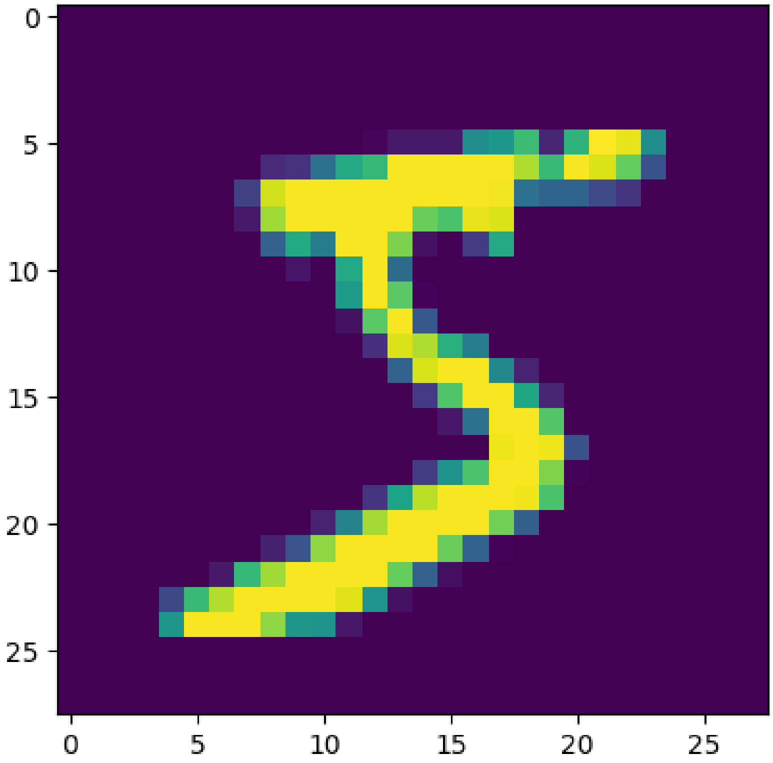
Rick Shen

Fall 2024

Machine Learning in Finance - Homework 1 – Due 09.10.2024

Textbook reading:
Chapters 1 and 2

- Google Colab account
- Import code into your Colab account from https://github.com/deepintomlf/mlfbook.git
- Install python packages including
  - Pandas
  - Numpy
  - Keras and Tensorflow
- A local installation of Anaconda and Jupyter notebook
- Upload section2.2_from_regression_to_classification.ipynb to colab and execute the code
- If the following line in the code shows an error message: model = LogisticRegression(random_state=0, solver='saga')
  - What is that error message and why does it matter?
  - How to you modify the code so the code runs to completion?



[h]

Figure 1: The first image from the training set

The code appears to do the following:

1. It loads the ‘MNIST’ dataset, which contains a training set of 60000 samples, each with a 78×78 pixle image of an integer number between 0 to 9.

2. The dataset also contains a testing set of size 10000, which is used to test the model.

3. It chooses the number 8, and aim to use logistic regression technique to tell if an image displays that number.

4. It uses several criterial to assess the model’s goodness of fit such as ROC curve.

5. Towards the end, it uses advanced techniques such as ensemble modeling.

When I first ran the code as-is, I’d get a warning message.

```
    from sklearn.linear_model import LogisticRegression

    model = LogisticRegression(random_state=0, solver='lbfgs')

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/
                                _logistic.py:940:
                                ConvergenceWarning: lbfgs
                                failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

It appears as if the regression did not converge.
Upon reading the document from Scikit-Learn, I had learned that scaling the predictors can improve the performance of the regression model. So I modified the code by normalizing the X values.

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)  # normalize X
                                values from training set
X_test_scaled = scaler.transform(X_test)    # normalize X
                                values from testing set

model = LogisticRegression(random_state=0, solver='lbfgs')
clf = model.fit(X_train_scaled, Y_train)

# Predict labels
y_train_est = clf.predict(X_train_scaled)
y_train_prob_est = clf.predict_proba(X_train_scaled)

# Predict probabilities
y_test_est = clf.predict(X_test_scaled)
y_test_prob_est = clf.predict_proba(X_test_scaled)
```

By doing so, no only did the warning message disappeared, the ROC-AUC value also increased slightly, indicating that indeed scaling the predictor can improve performance.

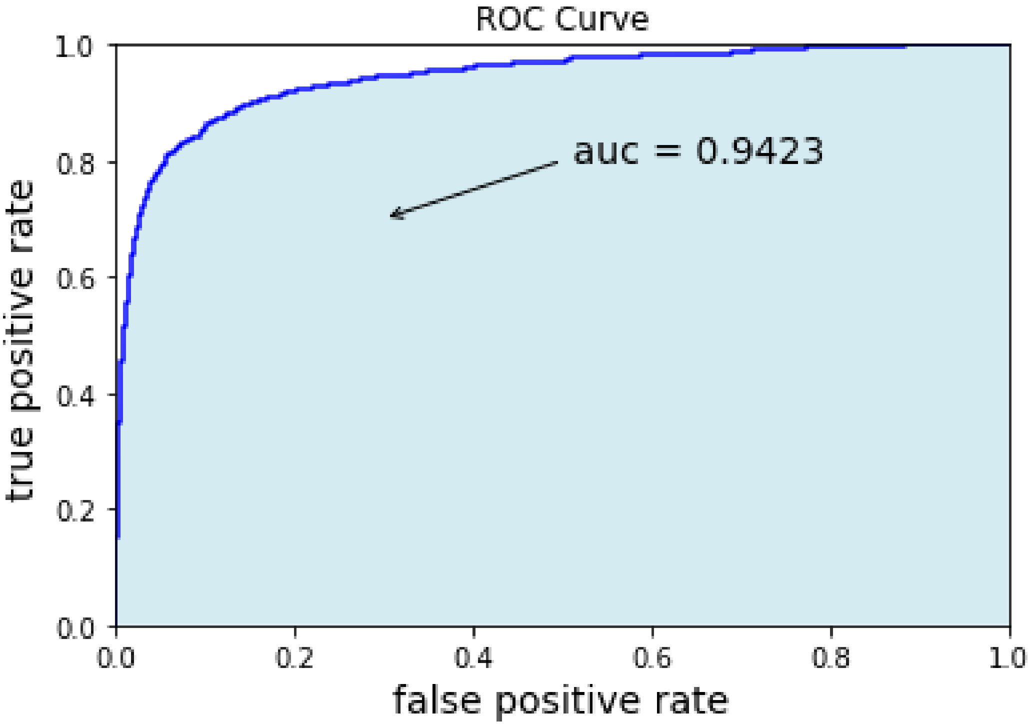


Figure 2: Area under the curve before scaling

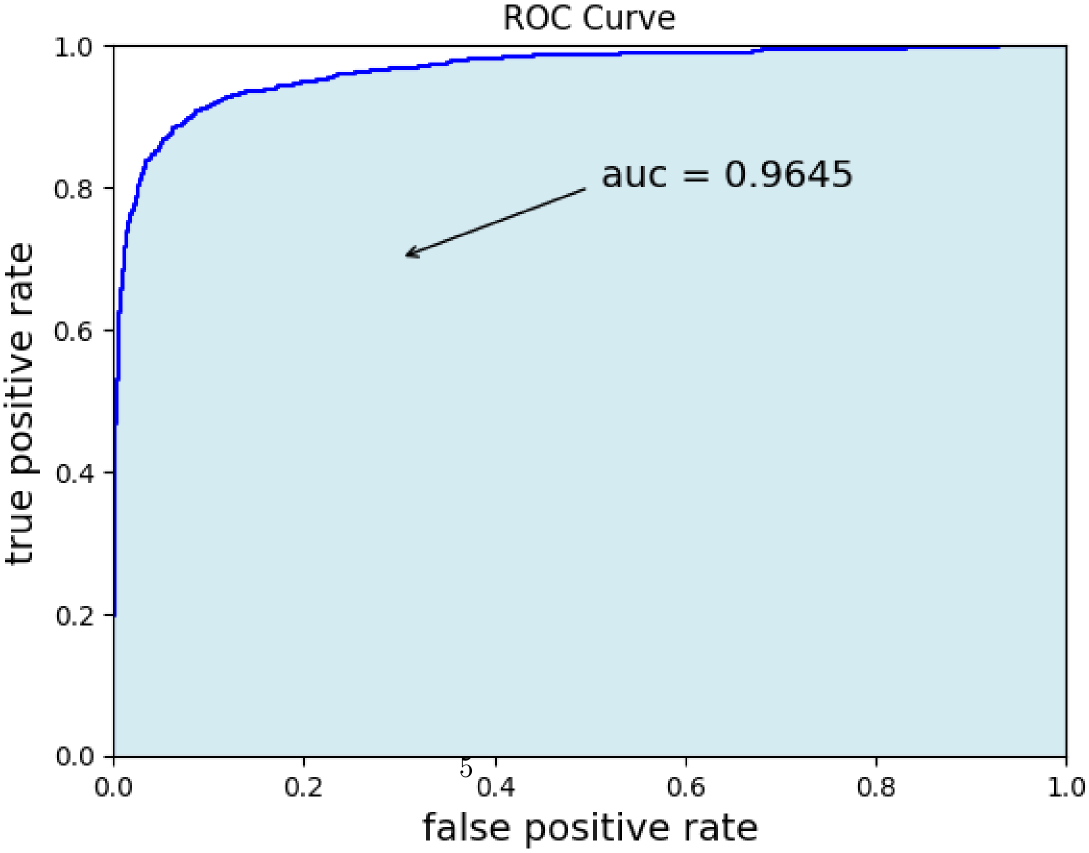


Figure 3: Area under the curve after scaling