Unit 11 - Risky Business

Credit Risk

Background

Mortgages, student and auto loans, and debt consolidation are just a few examples of credit and loans that people seek online. Peer-to-peer lending services such as Loans Canada and Mogo let investors loan people money without using a bank. However, because investors always want to mitigate risk, a client has asked that you help them predict credit risk with machine learning techniques.

In this assignment you will build and evaluate several machine learning models to predict credit risk using data you'd typically see from peer-to-peer lending services. Credit risk is an inherently imbalanced classification problem (the number of good loans is much larger than the number of at-risk loans), so you will need to employ different techniques for training and evaluating models with imbalanced classes. You will use the imbalanced-learn and Scikit-learn libraries to build and evaluate models using the two following techniques:

1. Resampling
2. Ensemble Learning

Files

Resampling Starter Notebook

Ensemble Starter Notebook

Lending Club Loans Data

Instructions

Resampling

Use the imbalanced learn library to resample the LendingClub data and build and evaluate logistic regression classifiers using the resampled data.

To begin:

1. Read the CSV into a DataFrame.
2. Split the data into Training and Testing sets.
3. Scale the training and testing data using the `StandardScaler` from `sklearn.preprocessing`.
4. Use the provided code to run a Simple Logistic Regression:

- Fit the `logistic regression classifier`.
- Calculate the `balanced accuracy score`.
- Display the `confusion matrix`.
- Print the `imbalanced classification report`.

Next you will:

1. Oversample the data using the `Naive Random Oversampler` and `SMOTE` algorithms.
2. Undersample the data using the `Cluster Centroids` algorithm.
3. Over- and undersample using a combination `SMOTEENN` algorithm.

For each of the above, you will need to:

1. Train a `logistic regression classifier` from `sklearn.linear_model` using the resampled data.
2. Calculate the `balanced accuracy score` from `sklearn.metrics`.
3. Display the `confusion matrix` from `sklearn.metrics`.
4. Print the `imbalanced classification report` from `imblearn.metrics`.

Use the above to answer the following questions:

- Which model had the best balanced accuracy score? >
- Which model had the best recall score? >
- Which model had the best geometric mean score?

Ensemble Learning

In this section, you will train and compare two different ensemble classifiers to predict loan risk and evaluate each model. You will use the Balanced Random Forest Classifier and the Easy Ensemble Classifier. Refer to the documentation for each of these to read about the models and see examples of the code.

To begin:

1. Read the data into a DataFrame using the provided starter code.
2. Split the data into training and testing sets.
3. Scale the training and testing data using the `StandardScaler` from `sklearn.preprocessing`.

Then, complete the following steps for each model:

1. Train the model using the quarterly data from LendingClub provided in the `Resource` folder.
2. Calculate the balanced accuracy score from `sklearn.metrics`.
3. Display the confusion matrix from `sklearn.metrics`.

4. Generate a classification report using the `imbalanced_classification_report` from imbalanced learn.
5. For the balanced random forest classifier only, print the feature importance sorted in descending order (most important feature to least important) along with the feature score.

Use the above to answer the following questions:

- Which model had the best balanced accuracy score?
- Which model had the best recall score?
- Which model had the best geometric mean score?
- What are the top three features?

Hints and Considerations

Use the quarterly data from the LendingClub data provided in the `Resources` folder. Keep the file in the zipped format and use the starter code to read the file.

Refer to the imbalanced-learn and scikit-learn official documentation for help with training the models. Remember that these models all use the model->fit->predict API.

For the ensemble learners, use 100 estimators for both models.

Submission

- Create Jupyter notebooks for the homework and host the notebooks on GitHub.
- Include a markdown that summarizes your homework and include this report in your GitHub repository.
- Submit the link to your GitHub project to Bootcamp Spot.

Requirements

Resampling (20 points)

To receive all points, your code must:

- Oversample the data using the Naive Random Oversampler and SMOTE algorithms. (5 points)
- Undersample the data using the Cluster Centroids algorithm. (5 points)

- Oversample and undersample the data using the SMOTEENN algorithim. (5 points)
- Generate the Balance Accuracy Score, Confusion Matrix and Classification Report for all of the above methods. (5 points) #### Classification Analysis - Resampling (15 points)

To receive all points, your code must:

- Determine which resampling model has the Best Balanced Accuracy Score. (5 points)
- Determine which resampling model has the Best Recall Score Model. (5 points)
- Determine which resampling model has the Best Geometric Mean Score. (5 points)

Ensemble Learning (20 points)

To receive all points, your code must:

- Train the Balanced Random Forest and Easy ensemble Classifiers using the Quarterly Data. (4 points)
- Calculate the Balance Accuracy Score using sklearn.metrics. (4 points)
- Print the Confusion Matrix using sklearn.metrics. (4 points)
- Generate the Classification Report using the `imbalanced_classification_report` from imbalanced learn. (4 points)
- Print the Feature Importance with the Feature Score, sorted in descending order, for the Balanced Random Forest Classifier. (4 points)

Classification Analysis - Ensemble Learning (15 points)

To receive all points, your code must:

- Determine which ensemble model has the Best Balanced Accuracy Score. (4 points)
- Determine which ensemble model has the Best Recall Score. (4 points)
- Determine which ensemble model has the Best Geometric Mean Score. (4 points)
- Determine the Top Three Features. (3 points)

Coding Conventions and Formatting (10 points)

To receive all points, your code must:

- Place imports at the beginning of the file, just after any module comments and docstrings and before module globals and constants. (3 points)
- Name functions and variables with lowercase characters and with words separated by underscores. (2 points)

- Follow Don't Repeat Yourself (DRY) principles by creating maintainable and reusable code. (3 points)
- Use concise logic and creative engineering where possible. (2 points)

Deployment and Submission (10 points)

To receive all points, you must:

- Submit a link to a GitHub repository that's cloned to your local machine and contains your files. (5 points)
- Include appropriate commit messages in your files. (5 points)

Code Comments (10 points)

To receive all points, your code must:

- Be well commented with concise, relevant notes that other developers can understand. (10 points)
