ITCS4156

Identifying safe loans with Decision Trees

Description
The LendingClub is a peer-to-peer leading company that directly connects borrowers and potential lenders/investors. In this assignment, you will build a classification model to predict whether or not a loan provided by LendingClub is likely to default.

You will use data from the LendingClub to predict whether a loan will be paid off in full or the loan will be charged off and possibly go into default. In this assignment you will:
- Do some feature engineering.
- Train a decision-tree on the LendingClub dataset.
- Visualize the tree.
- Predict whether a loan will default along with prediction probabilities (on a validation set).
- Train a complex tree model and compare it to simple tree model.
- Quantify the error cost of the model

Requirements
- Download the LendingClub dataset in CSV format from the assignment description on Canvas.

Part 1: Dataset Exploration

1. Print out the column names to see what features are in this dataset.

2. What is the number of attributes in the dataset?

3. What is the number of observations?

4. The target column (label column) of the dataset that we are interested in is called bad_loans. In this column **1** means a risky (bad) loan **0** means a safe loan.

5. Reassign the target to be:
   - **+1** as a safe loan
   - **-1** as a risky (bad) loan

6. Explore the distribution of the column safe_loans. This gives you a sense of how many safe and risky loans are present in the dataset. Print out the percentage of safe loans and risky loans.

7. Does this distribution (split) make the problem of identifying risky loans challenging? Explain your answer.

Part 2: Feature Selection

1. In this assignment, you will be using a subset of features (categorical and numerical). You can refer to the LendingClub website for more details about this dataset and these features. The features we will be using are:
   - grade # grade of the loan
   - sub_grade # sub-grade of the loan
   - short_emp # one year or less of employment
   - emp_length_num # number of years of employment
   - home_ownership # home_ownership status: own, mortgage or rent
   - dti # debt to income ratio
   - purpose # the purpose of the loan
   - term # the term of the loan
   - last_delinq_none # has borrower had a delinquency
   - last_major_derog_none # has borrower had 90 day or worse rating
   - revol_util # percent of available credit being used
   - total_rec_late_fee # total late fees received to day

Extract these feature columns and target column from the dataset. What remains now is a subset of features and the target that you will use for training the model.

Part 3: Building a Decision Trees Classifier

In this part you will build a Decision Trees classifier and train it using a subset of the data.

1. Balance data distribution among the two classes.

2. For scikit-learn's decision tree implementation, it requires numerical values for its data matrix. This means you will have to turn categorical variables into binary features via one-hot encoding. More info on this can be found at http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

3. Split data into training and validation using an 80/20 split. Call the training and validation sets **train_data** and **validation_data**, respectively.

4. Build a decision tree classifier using the built-in scikit learn decision tree learner (sklearn.tree.DecisionTreeClassifier) to create a loan prediction model on the training data. To do this, you will need to import **sklearn**, **sklearn.tree,** and **numpy**. Call this model **decision_tree_model**. Refer to http://scikit-learn.org/stable/modules/tree.html - decision-trees for an example on Decision Trees in scikit-learn.

5. Train a tree using **max_depth=2**. Call this model **small_model**.

Part 4: (Optional) Visualizing the learned model

For this optional section, you can see what the small learned tree looks like. You can use the Graphviz package to perform this step.
More info on this can be found at
http://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html

1. Visualize **small_model**.

2. You can also use the attributes of the sklearn.tree.DecisionTreeClassifier to explore the model. Which feature is the root of the **decision_tree_model**?

Part 5: Making predictions and exploring predictions

In this part you will use the model you in part 2 to make predictions on the likelihood of a LendingClub loan to default.

1. Consider two positive and two negative examples **from the validation set** and see what the model predicts. Use the **decision_tree_model** to predict whether or not these 4 sample loans are classified as a **safe loan**. (If you are using scikit-learn, you can use the **.predict()** method)

2. What percentage of the predictions on these 4 samples did **decision_tree_model** get correct?

3. What is the probability (according to **decision_tree_model**) of a loan being classified as **safe**? (If you are using scikit-learn, you can use the **.predict_proba()** method)

4. Which loan has the highest probability of being classified as a **safe loan**?

5. Can you verify that for all the predictions with probability >= 0.5, the model predicted the label **+1**?

6. Using the **small_model** what is the probability of these four loans being classified as **safe**?

7. How do these predictions compare to the predictions made by the **decision_tree_model**?

Part 6: Evaluating accuracy of the decision tree model

In this part you will examine the performance of the decision tree models you trained in the previous part.
Accuracy is defined as follows:

$$Accuracy = \frac{\#\ correctly\ classified\ data\ points}{\#\ total\ data\ points}$$

1. Evaluate the accuracy of **small_model** and **decision_tree_model** on the training data. (If you are using scikit-learn, you can use the **.score()** method)

2. Now, evaluate the accuracy of the **small_model** and **decision_tree_model** on the entire **validation_data**, not just the 4 samples considered in part 5.

3. What is the accuracy of **decision_tree_model** on the validation set, rounded to the nearest .01?

Part 7: A complex decision tree model

For this part you need to train a large decision tree with **max_depth**=10. This will allow the learned tree to become deeper, and result in a more complex model.

1. Using sklearn.tree.DecisionTreeClassifier, train a decision tree with maximum depth = 10. Call this model **big_model**.

2. Evaluate the accuracy of **big_model** on the training set and validation set.

3. How does the performance of the **big_model** compare to the performance of the **decision_tree_model** on the training set?

4. How does the performance of the **big_model** compare to the performance of the **decision_tree_model** on the validation set?

5. Which model would you recommend as a final predictor for whether or not a loan provided by LendingClub is likely to default? Explain your recommendation.

Part 8: Quantifying the cost of mistakes

Every mistake the model makes in predicting whether a loan will default or not costs money. In this section, you will try and quantify the cost each mistake made by the Decision Trees model you built.

Assume the following:

- **False negatives**: Loans that were actually safe but were predicted to be risky. This results in an opportunity cost of losing a loan that would have otherwise been accepted.
- **False positives**: Loans that were actually risky but were predicted to be safe. These are much more expensive because it results in a risky loan being given.
- **Correct predictions**: All correct predictions don't typically incur any cost.

To compute the cost of mistakes made by the model:

1. Compute the predictions made by the model.

2. Compute the number of false positives.

3. Compute the number of false negatives.

4. Compute the cost of mistakes made by the model by adding up the costs of true positives and false positives.

5. If we assume that each false negative costs $10,000, while a false positive costs $20,000. What is the total cost of mistakes made by the **decision_tree_model** on the **validation_data**?

Part 9: (Optional Bonus) Implement a Decision Trees Classifier.

In this part you can implement your own Decision Trees classifier and use it classify the LendingClub loans.

1. Implement a Decision Tress model. What assumptions have you made?

2. Train your model using the training dataset (**train_data**).

3. What is the depth of your Decision Tree model? Which feature is the root of your Decision Tree model?

4. Evaluate the accuracy of your model the training data.

5. Evaluate the accuracy of your model on the validation data (**validation_data)**.

6. Compare the performance of your model to the performance of the **decision_tree_model, small_model,** and **big_model** models you built in this assignment.

Assignment Submission

What to submit using Canvas (Email submissions will NOT be accepted):

1. **DecisionTrees_results.pdf** – A PDF document with your write up for the results (report) and answers to the questions in each part of the assignment.

2. **DecisionTrees.zip** - An archive of your entire programming project source code stored in a standard ZIP file. Make sure to include all packages and libraries used to run your programs if any.

3. **README.txt** –This file should detail all the files in your project archive, libraries and packages used and any special setup you have used in your programming environment.

4. **INFO.pdf** – PDF document with the following assignment information:
    a. Explanation of status and stopping point, if incomplete.
    b. Explanation of additional functions and analysis, if any.
    c. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?