1. **The LASSO and Boosting for Regression**

   (a) Download the Communities and Crime data[1] from `https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`. Use the first 1495 rows of data as the training set and the rest as the test set.

   (b) The data set has missing values. Use a data imputation technique to deal with the missing values in the data set.

   (c) Plot a correlation matrix for the features in the data set.

   (d) Calculate the Coefficient of Variation CV for each feature, where $CV = \frac{s}{m}$, in which s is sample variance and m is sample mean..

   (e) Pick $\lfloor\sqrt{128}\rfloor$ features with highest CV, and make scatter plots and box plots for them. Can you draw conclusions about significance of those features, just by the scatter plots?

   (f) Fit a linear model using least squares to the training set and report the test error.

   (g) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

   (h) Fit a LASSO model on the training set, with λ chosen by cross-validation. Report the test error obtained, along with a list of the variables selected by the model. Repeat with normalized features. Report the test error for both cases and compare them.

   (i) Fit a PCR model on the training set, with M (the number of principal components) chosen by cross-validation. Report the test error obtained.

   (j) In this section, we would like to fit a boosting tree to the data. As in classification trees, one can use any type of regression at each node to build a multivariate regression tree. Because the number of variables is large in this problem, one can use $\mathscr{L}_1$-penalized regression at each node. Such a tree is called $\mathscr{L}_1$ penalized gradient boosting tree. You can use XGBoost to fit the model tree. Determine α (the regularization term) using cross-validation.

2. **Tree-Based Methods**

   (a) Download the APS Failure data from: `https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks#` . The dataset contains a training set and a test set. The training set contains 60,000 rows, of which 1,000 belong to the positive class and 171 columns, of which one is the class column. All attributes are numeric.

   (b) Data Preparation

   This data set has *missing values*. When the number of data with missing values is significant, discarding them is not a good idea. [2]

[1] Question you may encounter: I tried opening the dataset and download it but the file is not readable. How to download the file? Just change .data to .csv. .

[2] In reality, wehn we have a model and we want to fill in missing values, we do not have access to training data, so we only use the statistics of test data to fill in the missing values. For simplicity, in this exercise, you first fill in the missing values and then split your data to training and test sets.

The data description mentions some features are nonpredictive. Ignore those features.

i. Research what types of techniques are usually used for dealing with data with missing values.[3] Pick at least one of them and apply it to this data in the next steps.[4]

ii. For each of the 170 features, calculate the coefficient of variation $CV = \frac{s}{m}$, where s is sample variance and m is sample mean.

iii. Plot a correlation matrix for your features using pandas or any other tool.

iv. Pick $\lfloor\sqrt{170}\rfloor$ features with highest CV, and make scatter plots and box plots for them, similar to those on p. 129 of ISLR. Can you draw conclusions about significance of those features, just by the scatter plots? This does not mean that you will only use those features in the following questions. We picked them only for visualization.

v. Determine the number of positive and negative data. Is this data set imbalanced?

(c) Train a random forest to classify the data set. Do NOT compensate for class imbalance in the data set. Calculate the confusion matrix, ROC, AUC, and misclassification for training and test sets and report them (You may use pROC package). Calculate Out of Bag error estimate for your random forset and compare it to the test error.

(d) Research how class imbalance is addressed in random forests. Compensate for class imbalance in your random forest and repeat 2c. Compare the results with those of 2c.

(e) Model Trees

In the case of a univariate tree, only one input dimension is used at a tree split. In a multivariate tree, or model tree, at a decision node all input dimensions can be used and thus it is more general. In univariate classification trees, majority polling is used at each node to determine the split of that node as the decision rule. In model trees, a (linear) model that relies on all of the variables is used to determine the split of that node (i.e. instead of using $X_j > s$ as the decision rule, one has $\sum_j \beta_j X_j > s.$ as the decision rule). Alternatively, in a regression tree, instead of using average in the region associated with each node, a linear regression model is used to determine the value associated with that node.

One of the methods that can be used at each node is Logistic Regression. One can use scikit learn to call Weka[5] to train Logistic Model Trees for classification. Train Logistic Model Trees for the APS data set without compensation for class imbalance. Use one of 5 fold, 10 fold, and leave-one-out cross validation methods to estimate the error of your trained model and compare it with the test error. Report the Confusion Matrix, ROC, and AUC for training and test sets.

[3] They are called data imputation techniques.

[4] You are welcome to test more than one method.

[5] `http://fracpete.github.io/python-weka-wrapper/install.html.` may help.

(f) Use SMOTE (Synthetic Minority Over-sampling Technique) to pre-process your data to compensate for class imbalance. Train a Logistic Model Tree using the pre-processed data and repeat 2e. Do not forget that there is a right and a wrong way of cross validation here. Compare the uncompensated case with SMOTE.

3. ISLR 6.8.3
4. ISLR, 6.8.5
5. ISLR 8.4.5
6. ISLR 9.7.3
7. Extra Practice: ISLR 5.4.2, 6.8.4, 8.4.4, 9.7.2

Appendix

Weka for Mac users:

1. Download JDK 9 from `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
2. Add environment variables in Terminal using : `vi~/.bash_profile`
   (a) export `JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-9.0.4.jdk/Contents/Home`
   (b) export `PATH=$JAVA_HOME/bin:$PATH`
3. Restart Terminal
4. Get brew (package installer for Mac, if you don't have it) and install python (not necessary)
5. brew install pkg-config
6. brew install graphviz
7. pip install javabridge
8. pip install python-weka-wrapper

And you should be able to use WEKA in your Jupyter Notebooks