# ECE356 F22 – Lab 4 – Data Mining

Due date: December 6th

**Group work policy:** In this lab, students are expected to work in groups no larger than 3. Groups of 1 or 2 are fine.

**Objective:**

- develop a working knowledge of classification using decision trees
- gain familiarity with data mining tools (MATLAB, Python)
- practice critical thinking skills

**Data set:** In this lab, you will perform data mining computations over the Sean Lahman baseball statistics data set: https://www.seanlahman.com/baseball-archive/statistics/. This data set comprises 28 tables (19 core, 8 contributed, and 1 "upstream"). Some tables have over 100,000 rows, and some have more than 20 columns. The data files, in CSV format, are hosted on GitHub: https://github.com/chadwickbureau/baseballdatabank/archive/refs/tags/v2022.2.zip. The schema is described in a companion document: https://www.seanlahman.com/files/database/readme2021.txt.

## 1 Classification Tasks

In this lab, you will analyze the Sean Lahman data set and construct classifiers to predict the values of certain target variables related to the nomination and induction of players into the Baseball Hall of Fame. In general, players in the database can be grouped into three categories:

1. inducted into the Baseball Hall of Fame (player appears in HallOfFame table and inducted = 'Y')
2. nominated for the Baseball Hall of Fame but not inducted (player appears in HallOfFame table and inducted = 'N')
3. not nominated for the Baseball Hall of Fame (player does not appear in HallOfFame table)

Using combinations of these three categories, we will formulate two prediction problems, as explained in more detail below. **The classifiers for these problems are to be constructed using input variables outside of the HallOfFame table.** For example, this means that the voting data (e.g., ballots, needed, and votes) cannot be used as inputs to the classifier. The HallOfFame table should only be used to assign the correct classification label during training.

**Task A:** Given an arbitrary player, predict whether the player will be nominated for the Baseball Hall of Fame. The classification label is True if a player belongs to category 1 or 2 above, and False otherwise.

**Task B:** Given a player who has been nominated for the Baseball Hall of Fame, predict whether the player will be inducted. The classification label is True if a player belongs to category 1 above, and False is the player belongs to category 2 above. Players in category 3 are not considered at all in this task.

## 2 Methodology

For a prediction problem, we start by identifying the features that will be used as input variables. In *feature selection*, features in the data set that may be suitable for the classification problem are chosen. Features that are redundant, or irrelevant to the problem, are dropped at this stage.

Since we are dealing with a relational database as the input data set, each feature is an attribute of some table. Thus, feature selection requires writing SQL code that joins the Player ID with other attributes that may be useful for prediction, as well as the correct classification label. The HallOfFame table should only be used to compute the latter. The output of the SQL query must be saved as human-readable CSV (comma-separate values) text file that can be imported into the data mining tool. Produce a separate file for Task A and Task B, and format each CSV file so that the player ID (string) is in the first column, and the classification label (Boolean) is in the last column. The CSV files must include a header row of the following form: $playerID, feature1, feature2, ...., featureN, class.$

After feature selection, you will **construct decision tree classifiers** using MATLAB or Python to solve the prediction problems defined earlier as Task A and Task B. To evaluate your decision trees, you will divide the data set into two separate parts: training set (80%) and the testing set (20%). The data mining tool will then build the decision tree model over the training set, and validate it using the testing set. The validation results will be summarized using a **confusion matrix**, which is the basis for computing correctness measures such as accuracy, recall, precision, and the F1 score. You will tune the decision tree by experimenting with different hyperparameters (e.g., impurity measure, maximum depth, etc.) to optimize its predictive power, and reflect on your design choices.

## 3 Deliverables

The expected submission for this lab is a collection of files, to be submitted electronically using the Lab 4 dropbox in LEARN. Either submit a collection of individual files, or submit a single zip archive with all your files. Each deliverable is described in more detail below.

1. **Report**
   Write a report on your findings, and submit it as a single PDF file that includes all supporting tables and figures. The report should be approximately 2000 words long, and include the following content:
   (a) Include the names, Waterloo emails, and student numbers of all group members on the title page.
   (b) For each classification task, compute the total number of instances in the data set, as well as the number of instances in each class and the corresponding proportion. Present the SQL queries used to compute these numbers.
   (c) Describe any data cleansing issues encountered in the data set, and their solutions. Be creative.
   (d) Explain your initial selection of features from the database. Provide rationale for each feature you selected.
   (e) Explain how exactly you sampled the training and testing subsets. Justify your sampling strategy.
   (f) For each of the two prediction tasks defined earlier, report on the validation results. At minimum, present the confusion matrix as a table, and indicate the accuracy obtained. Justify any other correctness measures discussed. Why should one measure be preferred over another?
   (g) For each prediction problem, explain how you tuned the hyperparameters to obtain alternative solutions that achieve different trade-offs with respect to the complexity of the model (e.g., size of the tree) versus its predictive power (e.g., accuracy). Which of the features you selected initially are actually used by your alternative solutions? How did you address the problem of over-fitting?
   (h) For the models discussed in the previous point, present illustrations of the decision trees. The illustrations must show clearly the attributes used in different splits, as well a structural properties such as the depth, branching factor, and number of leaf nodes. You may copy and paste visualizations from the data mining tool.

(i) Check the report carefully for spelling and grammar mistakes. Polish your writing.

2. **SQL code**
   Submit the SQL scripts you used to export the selected features and classification labels for each prediction task into CSV files.

3. **CSV file**
   Submit the CSV files produced by your SQL scripts, one for each task. Use a correctly formatted header row, as explained earlier.

4. **MATLAB or Python code**
   Submit the MATLAB or Python code you use to construct your decision tree classifiers and output their validation results.

Your submission will be checked for completeness, and also assessed for quality. Highest grades will be awarded to submissions that are technically deep and well presented.

# 4 MATLAB reference material

| MATLAB function | Description |
|---|---|
| **MODEL = FITCTREE (TBL, Y)** | This function is used to train the Decision Trees model in MATLAB. It needs two arguments: TBL refers to the data with the actual data values, and Y are the classification labels, provided as a part of the training dataset. It returns a Decision Tree model that can be used when testing the algorithm. For further details, you can refer to the MathWorks® website: https://www.mathworks.com/help/stats/fitctree.html. |
| **YP = PREDICT (MODEL, DATA)** | This MATLAB function can be used to predict the output of a trained model. The two arguments are the MODEL which is the model you would have achieved while training the data, and DATA refers to the test data on which the model needs to run. The ground truth of the testing sample is thus only used for evaluation purposes when you will be calculating accuracy of the results. For further details, you can refer to the MathWorks® website: https://www.mathworks.com/help/stats/compactclassificationtree.predict.html. |

Table 1: MATLAB functions for constructing decision tree classifiers.

| MATLAB function | Description |
|---|---|
| **loss** | Calculates the classification error of the model generated |
| **confusionmat** | Indicates how many labels were correctly classified and how many of them are inaccurate |
| **view** | Gives the textual and visual representation of the Decision tree model, including the split attribute, split condition and the path it takes in the tree (for example, left child or right child, etc) as a result of that condition. |

Table 2: MATLAB functions for validating and visualizing decision tree classifiers.

| MATLAB hyperparameters | Description |
|---|---|
| **MaxNumSplits** | The maximum number of branch node splits is MaxNumSplits per tree. Set a large value for MaxNumSplits to get a deep tree. |
| **MinLeafSize** | Each leaf has at least MinLeafSize observations. Set small values of MinLeafSize to get deep trees. The default is 1. |
| **MinParentSize** | Each branch node in the tree has at least MinParentSize observations. Set small values of MinParentSize to get deep trees. The default is 10 |

Table 3: MATLAB functions for tuning selected hyperparameters.

# 5 Python reference material

| Python Sklearn function | Description |
|---|---|
| **clf = DecisionTreeClassifier()**<br>**clf = clf.fit(X_train,y_train)** | This function is used to train the Decision Tree models in Python. It needs two arguments: X_train refers to the data with the actual data values, and y_train are the classification labels, provided as a part of the training dataset. It returns a Decision Tree model that can be used when testing the algorithm. For further details, you can refer to the scikit-learn website: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. |
| **y_pred = clf.predict(X_test)** | This function can be used to predict the output of a trained model. The argument X_test refers to the test data on which the model needs to run. The ground truth of the testing sample is thus only used for evaluation purposes when you will be calculating accuracy of the results. For an example you can refer to the datacamp tutrorial: website: https://www.datacamp.com/community/tutorials/decision-tree-classification-python. |

Table 4: Python Sklearn functions for constructing decision tree classifiers.

| Python Sklearn function | Description |
|---|---|
| **confusion_matrix(y_true, y_pred)** | Indicates how many labels were correctly classified and how many of them are inaccurate |
| **clf.plot_tree(clf)** | Gives the textual and visual representation of the Decision tree model, including the split attribute, split condition and the path it takes in the tree (for example, left child or right child, etc) as a result of that condition. |

Table 5: Python Sklearn functions for validating and visualizing decision tree classifiers.

| Python Sklearn hyperparameters | Description |
|---|---|
| **max_depth** | The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples |
| **min_samples_split** | The minimum number of samples required to split an internal node. |
| **min_samples_leaf** | The minimum number of samples required to be at a leaf node. |

Table 6: Python Sklearn functions for tuning selected hyperparameters.