**CSCE 623: Machine Learning**

**Spring 2020**

**HW5**

In this assignment, you will explore using random forests for regression. You will be working with the **Hitters** dataset in this assignment. You will be evaluated on your applications of techniques and methodology, as well as the evidence you present and conclusions you draw with respect to the models.

Your homework will be composed of an integrated code and report product using Jupyter Notebook. In your answers to written questions, even if the question asks for a single number or other form of short answer (such as yes/no or which is better: a or b) you must provide supporting information for your answer to obtain full credit. Use python to perform calculations or mathematical transformations or generate graphs and figures or other evidence that explain how you determined the answer. Each step listed below should correspond to code and/or markdown in your report file. Use numbered comments in your code and numbered text segments (headers) in markdown to help identify the location of your answer.

**Inspired by ILSR Chapter 8 Question 10: You will use Regression Trees to predict Salary in the Hitters dataset**

**Load and Preprocess the data**

1. (*Code provided*) Load the ILSR\_hitters.csv dataset using pandas. Then, using pandas methods:
   1. Remove the observations (rows) for which the salary information is unknown
   2. Drop the “NewLeague” feature using: Hitters = Hitters.drop(['NewLeague'],axis=1)
   3. Convert remaining categorical variables such as ‘League’ to 0-1 dummy variables. One way to do this is with pandas “.map”:  
      Hitters['League'] = Hitters['League'].map({'A': 0, 'N': 1})
2. The salaries in the dataset are indicated in $1000.00s of dollars. Remember to account for this when displaying and reporting results. In order to improve model fitting performance, you should log-transform the salaries using numpy.log10. Remember to account for this log-transformation by un-transforming when making predictions and presenting your results on the test set (convert back to real salary dollars when reporting these attributes.)
3. (*Code Provided*) Using sklearn.model\_selection.train\_test\_split with random\_state = 1, create a “non-test” set consisting of 200 observations and a test set consisting of the remaining observations. Sequester the test set until the performance reporting steps (9-11).

**Explore the data & make hypotheses**

1. Explore the data. Use plots and discuss relationships between available features and Salary. Consider using the seaborn package to facilitate your exploration – for example, make a heatmap plot of the correlation between each pair of features to help you decide which pairs of features to explore further with pairs plots or scatterplots. Make at least one hypothesis about which features will be useful in predicting salary.

**Train the model & Tune Hyperparameters using Cross-Validation**

1. Using sklearn k-fold split (sklearn.model\_selection.KFold), write code to set up a k-fold cross-validation with the goal of choosing the best hyperparameters for a random forest model (sklearn.ensemble.RandomForestRegressor) Select and provide rationale for your choice for n\_splits based on amount of data you have available in the non-test set. Your goal is to determine the best combination of two parameters: maximum tree depth (max\_depth), and the number of features to consider at each split (max\_features). The hyperparameter max\_depth of the trees should include integer values from 1 to 20, and your exploration over max\_features should include values from 1 to *p* (all features). You can decide whether to fix the number of trees (n\_estimators) or include it as a third hyperparameter to explore (it should start with at least 100 but you may want to explore higher values if you will tune this hyperparameter with cross-validation) – then explain whether you are selecting a specific value or tuning this value with cross validation. Since you will use a cross-validation wrapper to tune hyperparameters, set oob\_score to False in the initialization call to RandomForestRegressor. For each tuple of (max\_depth, max\_features), compute and collect the mean k-fold cross-validation MSE using predict().
2. Provide convincing visual evidence of the validation MSE performance (from step 5) as a function of max\_features and max\_depth (and n\_estimators if you chose to tune it). A good way to do this is to plot the error on a graph as a function of the two dimensions max\_depth and max\_features. Contour maps and heat maps would be valueable here.
3. Using code, determine, display visually, and report the values of these parameters with the lowest MSE. Discuss the minimum value of max\_features in light of the random forest recommendation for max\_features: sqrt(*p*) or *p*/3. Did your result agree with the general guidance on max features?
4. Using the best values of max\_features and max\_depth found with MSE, fit a new RandomForestRegressor model trained on **all** the **non-test** data.

**Reporting performance on the Test Set**

1. Using code, determine and report the quality of the model for predicting salary on the sequestered test set. Don’t forget to handle the log transformation you did in data preprocessing – your performance values should be based on real dollars (not log-transformed dollars).
2. Develop a scatterplot of the regression residuals: The figure’s x axis expresses the true dollar amount of salary, the figure’s y axis represents the prediction error (positive values mean underprediction, negative values mean overprediction, and y=0 would mean correct prediction). Discuss these residuals. Are they evenly distributed about y=0 through the range of possible true salaries? Do you see any patterns which would suggest true salaries for which prediction would be poor?
3. Using the model, report on variable importance - which variables appear to be the most important predictors in the model? Using the sklearn feature\_importance\_ attribute of the best fitted model, provide numerical and visual evidence to support your answer (make sure to sort your outputs by feature importance).