Problem Set #7

ECON 833, Prof. Jason DeBacker
Due Thursday, October 30, 1:15 p.m.

This problem set is designed to provide you with experience estimating regression trees in Python.[1]

Joe Biden was the 46th President of the United States. He has been the subject of many memes, attracted the attention of Leslie Knope (Parks and Recreation, TV sitcom), and experienced a brief surge in attention due to photos from his youth. The data file `biden.csv` contains a selection of variables from the 2008 American National Election Studies survey that allow you to test competing factors that may influence attitudes towards Joe Biden. The variables are coded as follows:

- `biden`: feeling thermometer ranging from 0 to 100. Feeling thermometers are a common metric in survey research used to gauge attitudes or feelings of warmth towards individuals and institutions. They range from 0-100, with 0 indicating extreme coldness and 100 indicating extreme warmth.
- `female`: =1 if respondent is female, =0 if respondent is male
- `age`: age of respondent in years, range from 18 to 93
- `dem`: =1 if respondent is a Democrat, =0 otherwise
- `rep`: =1 if respondent is a Republican, =0 otherwise
- `educ`: number of years of formal education completed by respondent, range from 0 to 17 with 17+ representing the first year of grad school and up.

Tasks

1. Split the data into a training set (70%) and a test set (30%) using the `sklearn.model_selection.train_test_split()` function with `random_state=25`. Setting the seed will guarantee you all get the same results. Use recursive binary splitting to fit a decision tree to the training data, with `biden` as the response variable and the other variables as predictors. Set the `max_depth=3` and `min_samples_leaf=5`. Plot the tree and interpret the results. What is the test MSE?

2. Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the decision tree from part (1). Tune the parameters `max_depth`, `min_samples_split`, and `min_samples_leaf`. Set `n_iter=100, n_jobs=-1, cv=5` for k = 5 k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared error'`. This last option will allow you to compare the MSE of the optimized tree (it will output the negative MSE) to the MSE calculated in part (1). Set your parameter distributions over which to test random combinations to the following:

```
from scipy.stats import randint as sp_randint
param_dist = {"max_depth": [3, 10],
              "min_samples_split": sp_randint(2, 20),
              "min_samples_leaf": sp_randint(2, 20)}
```

   Report your optimal tuning parameter values (use the `.best_params_` attribute of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` attribute of your `RandomizedSearchCV().fit(X, y))` results.

[1]This problem was originally created by Benjamin Soltoff as an application for the R programming language and adapted by Rick Evans.

3. Now tune the parameters of a RandomForest regression model on these data `sklearn.ensemble.RandomForestRegressor()`. Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the random forest regression model. Tune the parameters `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features`. Set `n_iter=100`, `n_jobs=-1`, `cv=5` for k = 5 k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your Random Forest parameter distributions over which to test random combinations to the following:

```
param_dist = {"n_estimators": [10, 200],
              "max_depth": [3, 10],
              "min_samples_split": sp_randint(2, 20),
              "min_samples_leaf": sp_randint(2, 20),
              "max_features": sp_randint(1, 5)}
```

   Report your optimal tuning parameter values (use the `.best_params_` attribute of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` attribute of your `RandomizedSearchCV().fit(X, y))` results.

Deliverable

You will turn in a Jupyter Notebook (where you have code to estimate the models and text answering the questions). You will put your notebook in the path `/CompEcon_Fall2025/ProblemSets/ProblemSet7` on your ProblemSets branch of your fork of the class repository.