Tree-based Algorithms and KNN Assignment

Adult-Income Prediction

Dataset link: Adult Dataset Income Prediction | Kaggle

The target of this assignment is to use tree-based algorithms and KNN to classify the income of the user whether it is above or below 50K.

Note that:

a. If you found that your dataset is imbalanced, please use class weights in your classifier (for the tree-based algorithm) while use under-sampling or over-sampling with KNN
b. The model with the best-performance should be evaluated according to the F1-score because we care here about both the precision and the recall of our classifier
c. The tree-based algorithms are supposed to obtain the same performance whether you scale the data or not (because it is based upon splitting on thresholds). In order to prove this to yourself, please try the decision tree 2 times: one time using scaled data, and other time without scaling the data, and both trials with the same hyper-parameters should obtain identical results

You are required to:

1- Visualize the data (all the columns)
2- Perform train-test split (80% – 20%)
3- Understand the features, handle any meaningless values in the columns, handle the missing values and the outliers
4- Perform data-preprocessing (standardize/min-max scale the numerical features, one-hot-encode or perform ordinal encoding to the categorical features)
5- Use KNN with the pre-processed data to classify all the users (above or below 50K):
   a. Try different p values in the Minkowski distance metric
   b. Try different N_neighbors
   c. Try 'uniform' and 'distance' options in the weights hyper-parameter

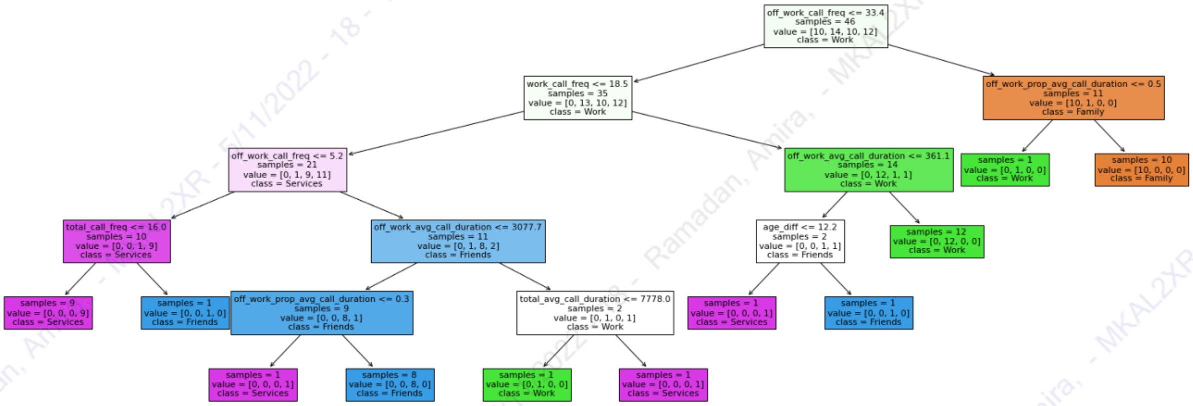
   Use different combinations of p and N_neighbors to find the best hyper-parameters that best classify the test data

6- Use Decision Tree to classify all the user (above or below 50K) with random_state = 2022:
   a. Try different values of max depth in the tree (start with max_depth = 3)
   b. Try different values of min_samples_split
   c. Try different values of min_samples_leaf
   d. Try different values of max_features (start with 3 then increase)
   e. Try different values of max_leaf_nodes
   f. Try different values of ccp_alpha (start with 0.001 then increase)

   Use different combinations of hyper-parameters to find the best hyper-parameters that classify the test data correctly

7- Use ‘tree’ library from Sklearn to visualize the decision tree as follows (you will need to search and read its documentation to be able to use it):

Please make sure that your tree is clear and understandable


off_work_call_freq <= 33.4 samples = 46 value = [10, 14, 10, 12] class = Work
work_call_freq <= 18.5 samples = 35 value = [0, 13, 10, 12] class = Work
off_work_prop_avg_call_duration <= 0.5 samples = 11 value = [10, 1, 0, 0] class = Family
off_work_call_freq <= 5.2 samples = 21 value = [0, 1, 9, 11] class = Services
off_work_avg_call_duration <= 361.1 samples = 14 value = [0, 12, 1, 1] class = Work
samples = 1 value = [0, 1, 0, 0] class = Work
samples = 10 value = [10, 0, 0, 0] class = Family
total_call_freq <= 16.0 samples = 10 value = [0, 0, 1, 9] class = Services
off_work_avg_call_duration <= 3077.7 samples = 11 value = [0, 1, 8, 2] class = Friends
age_diff <= 12.2 samples = 2 value = [0, 0, 1, 1] class = Friends
samples = 12 value = [0, 12, 0, 0] class = Work
samples = 9 value = [0, 0, 0, 9] class = Services
samples = 1 value = [0, 0, 1, 0] class = Friends
off_work_prop_avg_call_duration <= 0.3 samples = 9 value = [0, 0, 8, 1] class = Friends
total_avg_call_duration <= 7778.0 samples = 2 value = [0, 1, 0, 1] class = Work
samples = 1 value = [0, 0, 0, 1] class = Services
samples = 1 value = [0, 0, 1, 0] class = Friends
samples = 1 value = [0, 0, 0, 1] class = Services
samples = 8 value = [0, 0, 8, 0] class = Friends
samples = 1 value = [0, 1, 0, 0] class = Work
samples = 1 value = [0, 0, 0, 1] class = Services

8- Use Random Forest to classify all the user (above or below 50K) with random_state = 2022, try all the hyper-parameters of the decision tree in addition to:

a. n_estimators
b. Use Bootstrap = True
c. max_samples (start with 1000 and increase)

Use different combinations of hyper-parameters to find the best hyper-parameters that classify the test data correctly