Comp 551 - Project # 1 (Group 10)

Abstract:

In this project we made an analysis on two machine learning classification models, namely k-nearest neighbors and decision trees. We investigated the performance of these two models in predicting their respective target values. For the `adult` dataset the prediction was made on whether an adults' income exceeds $50K/yr depending on different input parameters such as age, sex, nationality, relationship, occupation etc. Whereas, for the `clothing fit` dataset the prediction was made on the size of a particular person depending on input parameters such as bra size, category, cup size, fit etc.

After fitting the models using the sklearn library with the respective datasets we used our custom written cross validation method to compare the performance of the two models on each of the datasets. Then we present the F1 scores of each of the models in terms of predicting the target variable and reach a conclusion that increasing the number of neighbors and max depth values improves model performance but only till a certain point.

Introduction:

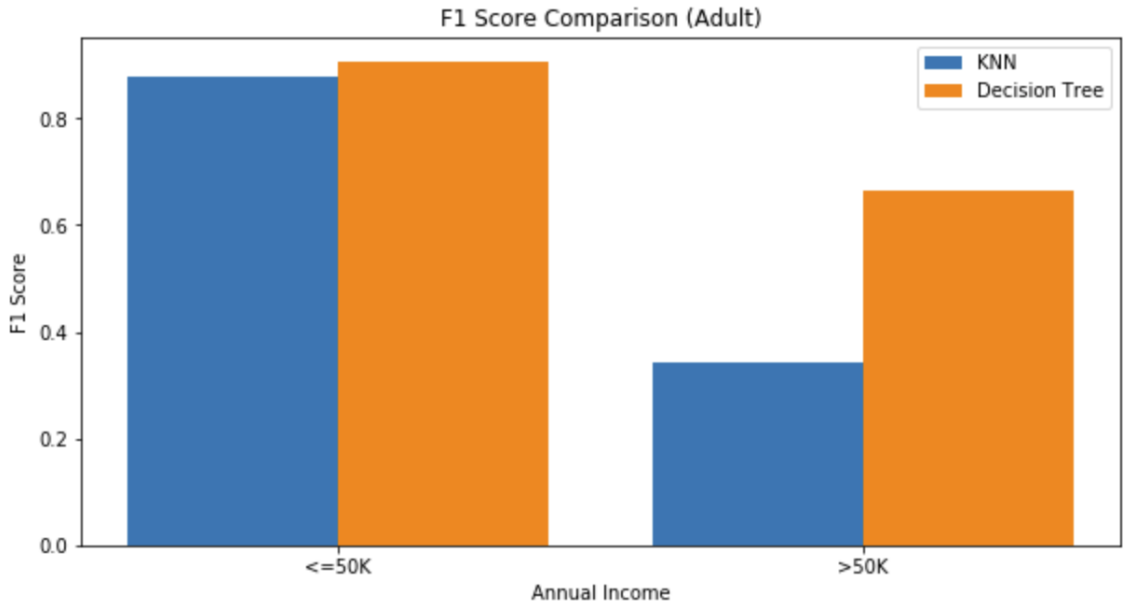
This report describes the steps we took to compare two different machine learning methods namely KNN and Decision Trees by implementing these on two datasets.
The first dataset had a size of 48845 instances. Each instance represents a person having features such as age, sex, nationality, relationships, Annual Income > or < = 50k etc. The target variable was the annual income which the models predicted using the dependent variables.
The second dataset had a size of 82790 instances. Each instance represents a person having features such as size, cup size, height etc. The target variable was the size of a person which the models predicted using the dependent variables.
We used the sklearn library to get the KNeighbors model and left the default value for p i.e p=2. p is the power parameter for the Minkowski metric and when p=2 it implies that we used the euclidean distance to get the nearest neighbors.
The comparison between the two models was easy to compare especially because both are non-parametric models. This article discusses how KNNs are more computationally costly due to a lack of a training phase and this is exactly what we observed during the execution of our experiment.

Datasets:

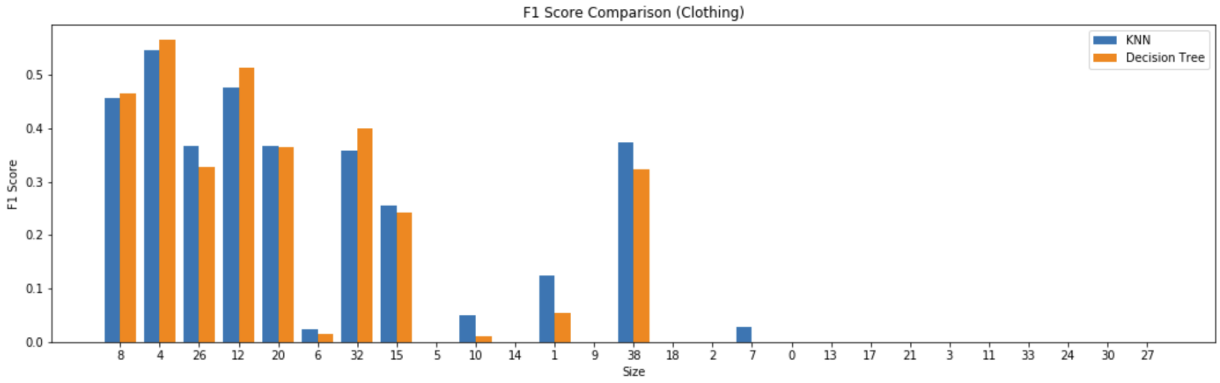
We are using two different datasets for the evaluations of our model: Adult and Clothing Fit.
From the Clothing Fit dataset we used the data associated with *modcloth*.
In order to clean the Adult dataset, we fixed the format of the "annual-income" column in the test set to match the format of the training set and removed all the rows with missing data. As a result of this cleaning the number of entries in our data went from 48845 to 45224.
In order to clean the Clothing Fit data we first had to run a bash script *fix_json_for_clothing_dataset.sh* which converts the file from a format where each row is a json object to a list of json objects for easier processing with Pandas. We then remove all the noisy columns which would not affect our size prediction such as *user_id* and *user_name*. As we want the size of this dataset to be similar to the size of the Adult dataset we can be lenient with removing rows with missing values for columns such as *height* and *length*. After the cleaning, the size of our data goes from 82790 to 54336.

Results:

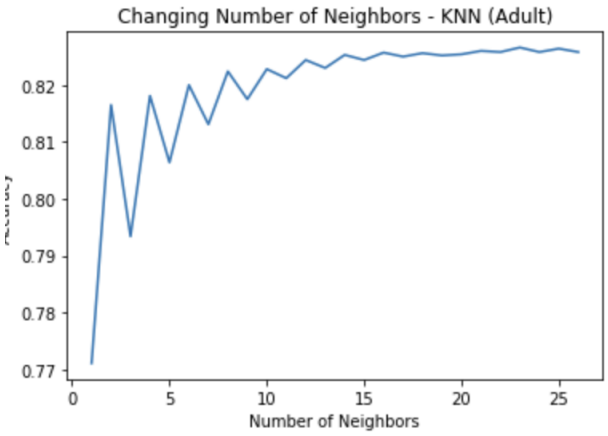
Comparing the performance of KNNs and Decision Trees using F1 scores we found that for the Adult dataset, Decision Trees seemed to outperform KNNs.There was a marginal difference in the classification accuracy of those of making more than 50K a year with decision trees as shown in the graph below.

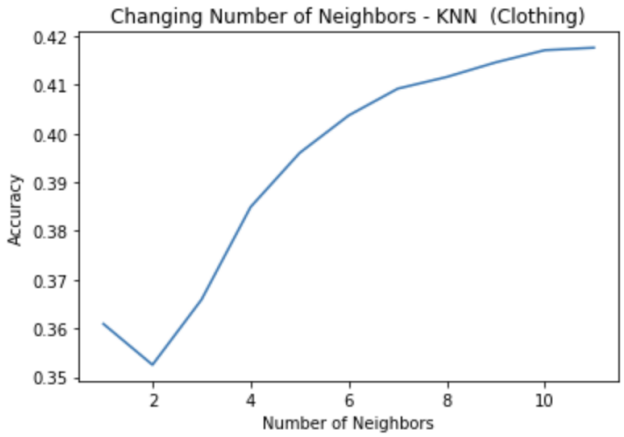


For the Clothing Fit dataset, both models performed poorly overall in determining an accurate size measurement. This may be due to the fact that this was not a binary classification. As shown below except for size 1 and 10, decision trees seemed to perform similarly or slightly better than KNNs. As a better measure of accuracy it might be useful to assign weights to each classification using the number of instances belonging to each size and determine an overall correct classification score.

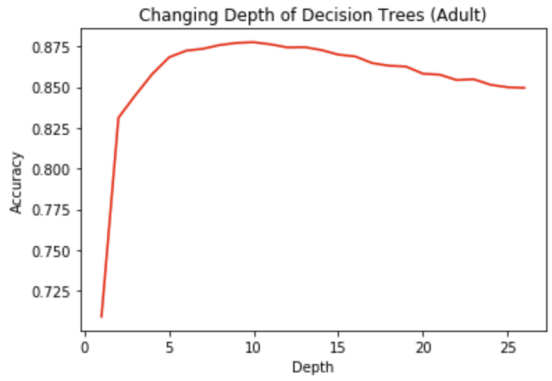


Change in hyperparameters led to great degree of change in the accuracy of our models. For KNNs you can observe this change for the Adult dataset on the left graph below and the Clothing Fit dataset on the right graph. For both datasets, increasing number of neighbors led to an increase in Accuracy with the accuracy converging after a certain point. If KNN was less computationally intensive we would be interested in looking at the point where the model starts overfitting and accuracy starts decreasing marginally.





Due to the running of Decision Trees being less computationally intensive we managed to iterate a higher number of times. For both Adult and Clothing Fit increasing max depth led to an increase in Accuracy up to a certain point. The best max depth was 10 for the Adult dataset and 9 for Clothing Fit.



Growing subsets of training/validation sets led to an overall increase in accuracy. This increase was not a perfect linear function as seen in the graphs in our *Jupyter Notebook.* Our suspicion is that this might be due to the randomization of the data as there is no guarantee that the data is independently identically distributed (IID).

Discussion and Conclusion:

The interesting take away from the comparison of the two models was how computationally intensive KNNs are to make predictions. The time taken for prediction differed by a great margin when compared to for Decision Trees. KNNs also make predictions based on numerical distance. This causes KNNs to perform better when the input space is only numerical data. On the other hand Decision Trees can process both numeric and nominal data.
We would be interested to see how accurate predictions could be if we used a more efficient Decision Tree model such as Random Forest which is a collection of several decision trees together.

Statements of Contributions:

Saad was responsible for cleaning both datasets and writing code for one hot encoding. Ibrahim was responsible for writing custom cross validation code. Sym was responsible for finding the second dataset and refactoring the code. All members contributed equally to the writing of the report.