CS-273P Machine Learning
Adult DataSet Project Report

Team Name -: Data Dragons

Student 1 -: Karan Khosla
Student ID -: 91725095
UCI Email -: khoslak@uci.edu

Student 2: Arvindh Kumar Chandran
Student ID -: 83002712
UCI Email -: arvindhc@uci.edu

Dataset description and preprocessing.

This adult dataset consists of 14 features to predict whether income for a person exceeds $50k/year or not. The 14 features are as follows-:
Age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week and native-country.

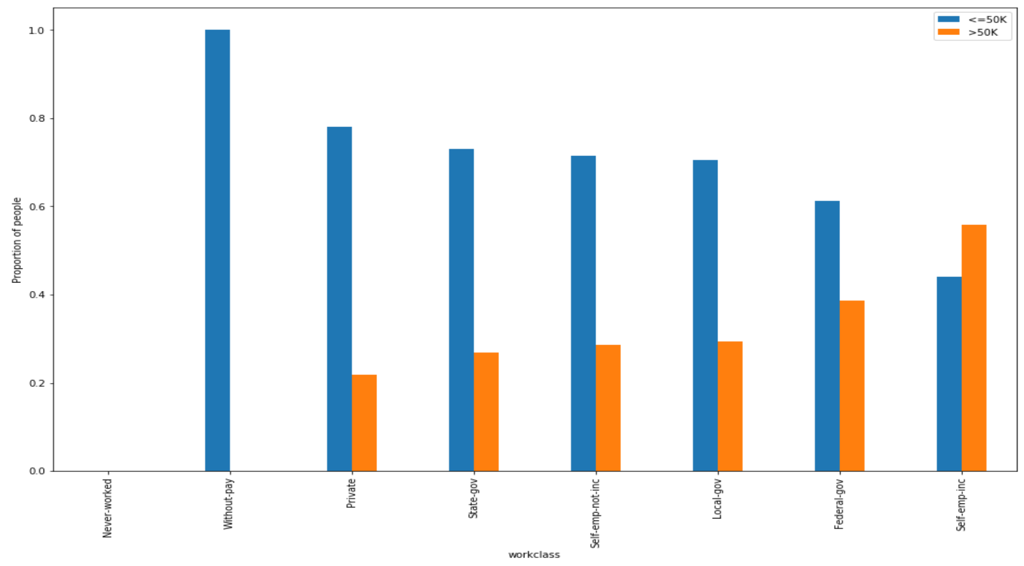
We load the data using the data_loader provided and find that there are 48842 instances of data, which is split into 32561 training data and 16281 test data by the provided data_loader function.
We perform a few preprocessing steps so that our data is suitable for training. These include -:

1. Converting the non-continuous columns to categorical in nature.
2. Removing the '.' from the end of the income column in test data and space from all data.
3. Figuring out the correlation between different columns.
4. Removing rows with missing values(?) in them and dropping the category class(?).
5. Note -:Removing a few rows from the data won't alter our training much.
6. Printing the statistics of the various columns of data to understand how it is skewed.
7. Mapping income less that 50K to -1 and income more than 50K to +1.

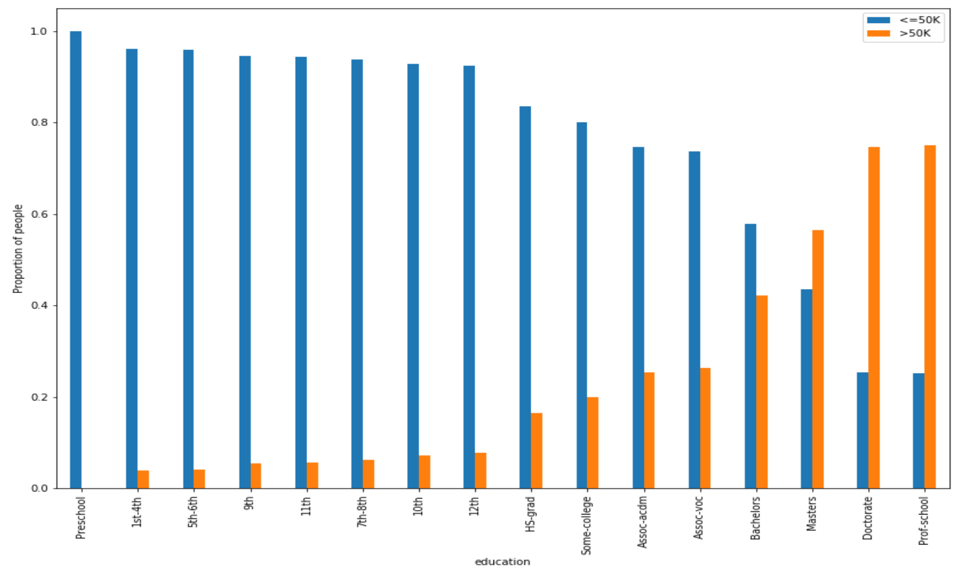
Data Visualization

To understand how the data is, now we move onto visualizing each of the categorical columns and try to find how each is(or isn't) related to the income being <=50K or >50K

1. Workclass vs income



2. **Education vs income**



We see that, as expected, the proportion of people earning more than 50K increases as people tend to attain higher levels of education. But, not until completing a Masters degree do people earning more than 50K exceed people earning less than 50K.
Similarly, we also check other columns (not posting the plot here due to space constraints).

3. **Occupation vs income**

Again, we see that people who have better 'sounding' jobs in general have a higher chance of earning more than others.

4. **Gender vs income**

It is seen that the proportion of men earning more than 50K is almost twice as women earning 50K income.

Data Preparation

Feature cleaning

1. As the education and the education_num is just a one to one mapping, we remove the education_num column from both the training set and the testing set. We do this to avoid the effect of this attribute on the models to be overstated.
2. We also remove the class variable from the train and the test data into separate arrays to be used in each models

One-hot encoding

This is used to convert the categorical columns into binary features. This does increase the dimensionality of the data, but the performance of the different models improves if we use this form of the data.

Normalization of train and test data

We do this to speed up our training process and so that each of the numerical columns of data have the same influence on the model training.

Models

In order to avoid overfitting the model on the training data and stabilize the bias-variance tradeoff, we use k-cross validation and try to find the values of the hyperparameters that give the best performance on the validation data(to prevent overfitting), if it is not overfitting on the training data. This we do to find the sweet spot of hyperparameters for our training data to get the best results for our test data.
We use the AUC metric to determine how well our k-cross validation runs for different hyperparameters and from the plots determine a set of hyperparameters on which the model does not overfit and still gives good results on our validation data.

On choosing these hyperparameters, we then train the model on the whole training dataset and determine how well it is trained by using four metrics -:

1. Accuracy - Percentage of right predictions
2. Precision - Out of those predicted positive, how many of them are actual positive.
3. Sensitivity - proportion of actual positives that are correctly identified as such.
4. Specificity - proportion of actual negatives that are correctly identified as such

These 4 parameters give us a good picture of our model. To compare our model performances, we use accuracy as our distinguishing metric.

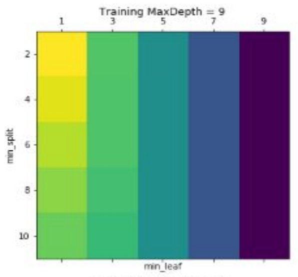
Gaussian Naive Bayes classifier

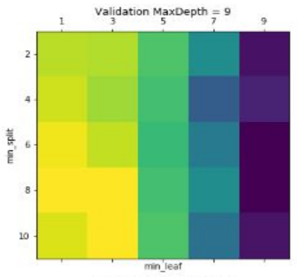
We are using Naive Bayes Classifier as our first model to get an idea of what to expect in terms of accuracy and performance going on with the other models. We get base results for the four performance metrics we are interested in as follows-:

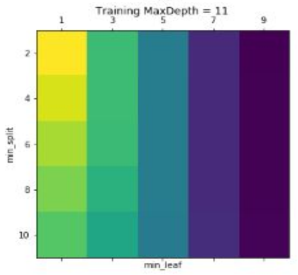
	accuracy	precision	sensitivity	specificity
Naive Bayes	0.753254	0.482222	0.979489	0.058649

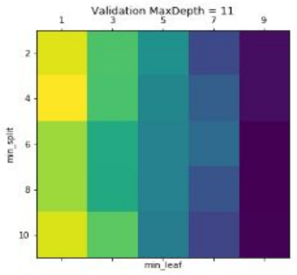
Decision Tree

This is a good choice for this dataset as the number of features in this dataset is low, hence this would give good performance both in terms of time taken to run, as well as our four metrics.
The hyperparameters to tweek here are max_depth, min_samples_split and the min_samples_leaf.
We plot the matrix show plots for each of the combinations of the 3 parameters with depth ranging from 7 to 15, min_sample_splits from 2 to 10, min_leaf from1 to 9(after conducting multiple runs with different parameters). We see the important plots below that were used to get a potential set of hyperparameters.
Left plots are on training data AUC and right on validation data AUC.









As seen from the plots above, we find three candidate values(where validation auc is great(yellow), given the training auc is not yellow as well) of our ideal hyperparameters, where the model does not overfit.
1. max_depth = 7, min_samples_split = 10, min_samples_leaf = 1
2. max_depth = 9, min_samples_split = 8, min_samples_leaf = 1
3. max_depth = 11, min_samples_split = 4, min_samples_leaf = 1

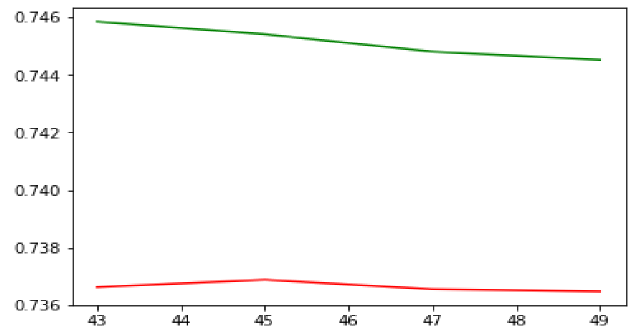
Training our Decision Tree model using our whole training data, we get the best accuracy when
1. max_depth : 9, min_samples_split : 8 , min_samples_leaf : 1
For this set of hyperparameters, we get the following values of our 4 performance metrics -:

	accuracy	precision	sensitivity	specificity
Decision Tree	0.853984	0.766607	0.942165	0.583243

We see that the decision tree performs much better than the gaussian naive bayes model.

k-Nearest Neighbor

The only hyperparameter to consider here is the number of nearest neighbors. If the number of neighbors is low, the model overfits and gives bad accuracy. So, we choose a range of 43 to 50 neighbors as our potential hyperparameters(by doing no of experiments of performance vs runtime of our model).



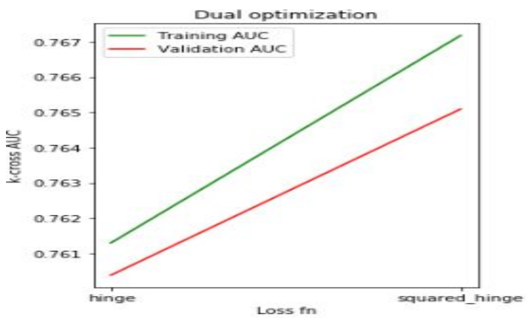
From the plot above, we determine that our model does not overfit if our parameter value is 49. Training this model on the n_neigh parameter value = 49, we get the following values of our metrics

	accuracy	precision	sensitivity	specificity
kNN	0.807304	0.812696	0.978961	0.28027

We see that kNN does not perform better than decision tree, but still better than Naive Bayes.

Linear SVM classifier

The linear SVM classifier only runs for certain combinations of hyperparameters. If we are using dual optimization, then penalty = l2 and we check for loss function. In primal optimization,loss = squared_loss. So, we check whether 'l1' or 'l2' gives better results.



Getting higher auc value for validation when we regularize with respect to l2 and dual problem(with squared hinge). Attempting to use the same configuration on the entire dataset now and calculate its performance.

	accuracy	precision	sensitivity	specificity
Linear SVM	0.807304	0.812696	0.978961	0.28027

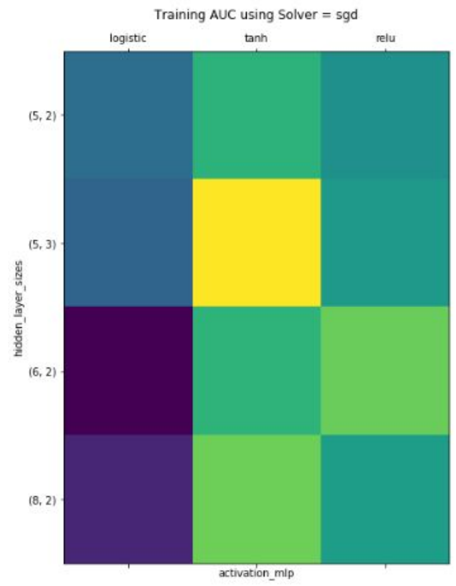
Still not the best performance for this dataset.

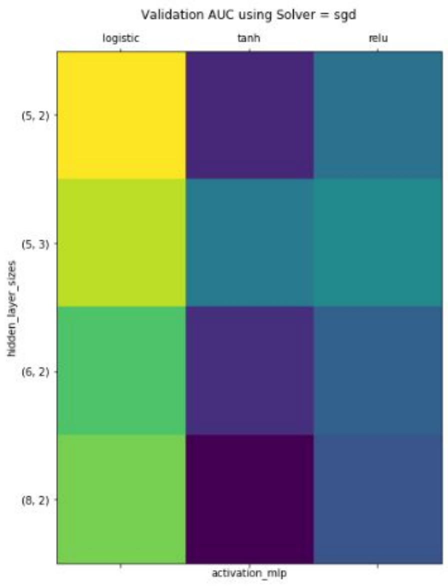
Multi-Level Perceptrons

We choose neural network because they can learn arbitrary boundaries as compared to decision trees. Hyperparameters considered here are hidden_layer_sizes, activation function and solver.

As seen from the plots below, we find three candidate values of our ideal hyperparameters

1. solver = sgd, hidden_layer_sizes = (5,2), activation = logistic
2. solver = sgd, hidden_layer_sizes = (5,3), activation = logistic
3. solver = adam, hidden_layer_sizes = (8,2), activation = logistic





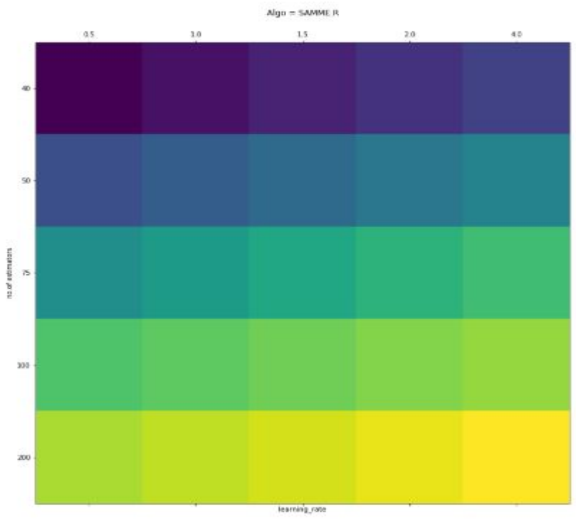
Out of these three, we find that max accuracy is when and the performance is

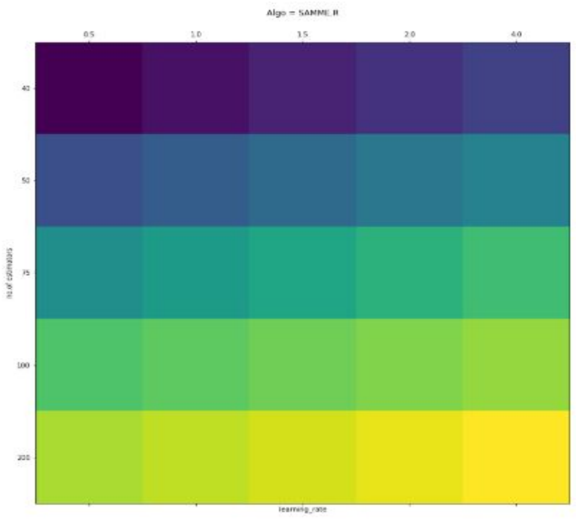
1. Solver : sgd, hidden_layer_sizes: (5, 2) ,activation: logistic

	accuracy	precision	sensitivity	specificity
MLP	0.853918	0.768049	0.94287	0.580811

Ensemble Model - AdaBoost

We also see the performance of ensemble models on our data to see whether they would give a better performance than any single model would. For the AdaBoost classifier, we are most interested in finding the best possible combination of the 3 hyperparameters, i.e., which algo out of discrete vs real boosting should be used, the no of estimators and the learning rate
We set the range for estimators from 40 to 200(not all values in between) and for learning rate from 0.5 to 4.0 and see at what combinations of the parameters, we get the best AUC results.





We find that (irrespective of the algo), when no of estimators is 200 and learning rate is 4.0 or 2.0, the model overfits and does not perform well on the test data. So, we tune our hyperparameters and select them as algo = SAMME.R, no of estimators = 200 and learning rate = 1.0 to get the best results.

	accuracy	precision	sensitivity	specificity
AdaBoost	0.863546	0.762193	0.934331	0.646216.

Model Evaluation and Conclusion

Final consolidated results -:

	accuracy	precision	sensitivity	specificity
AdaBoost	0.863546	0.762193	0.934331	0.646216
Decision Tree	0.853984	0.766607	0.942165	0.583243
MLPerceptron	0.853918	0.768049	0.942870	0.580811
kNN	0.807304	0.812696	0.978961	0.280270
Linear SVM	0.807304	0.812696	0.978961	0.280270
Naive Bayes	0.753254	0.482222	0.979489	0.058649

We see that AdaBoost out-perfoms all others when it comes to accuracy metric as well as specificity metric. As it is a pretty simplistic dataset, a relatively simple model like decision tree also does well, as compared to Multi-level perceptrons and ensemble models. However, AdaBoost avoids overfitting as it comprises of a bunch of learners, but is itself susceptible to noisy data and outliers.

Individual contributors

Work done by Karan Khosla

1. Data Cleaning - removing spaces and dots at the end of test income column
2. Data Cleaning - removing rows in test and train data that include missing values
3. Data Visualization - plotting workclass vs income and education vs income to understand how they affect income prediction.
4. Decision Trees
5. Multi-Level Perceptrons
6. Linear SVM
7. Evaluation and conclusion

Work done by Arvindh Kumar Chandra

1. Data preprocessing - loading data, assigning category names conversion to categorical columns.
2. Data preprocessing - summary analysis
3. Data preprocessing -removal of ? categories from columns
4. Data Visualization - plotting occupation vs income, gender vs income and race vs income to understand how they affect income prediction.
5. Gaussian Naive Bayes
6. kNN Classifier
7. AdaBoost classifier

References -:

Accuracy vs precision https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Specificity vs sensitivity
https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Need for normalization
https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

Normalizing dataframe
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

DataFrame plotting
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

Need of one-hot encoding
https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Classifier Comparison
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html