Project Report for Adult Data Income

Zahra Askarzadeh, Saumitra Prabhat Kadge, Elham Havvaei

Problem:

In this project we are using "Adult" data set in order to predict whether income exceeds $50K/yr based on census data. Hence, associated task for this project is classification. For this purpose, we use several classification models discussed in this course. The main libraries we use here are pandas, numpy, sklearn, matplotlib.pyplot.

Data Analysis:

This data set consist of 14 feature variables and one target value. Nine out of fourteen features are categorial and three of them, workclass, occupation and native-country, have missing data.

Missing values:

To deal with missing values, we combine three predictors: logistic regression, decision trees, and random forests. Predictions are based on the values of other features other than the one under consideration. If more than two predictors predict the same value, then it is considered as the final value. If no predictors predict the same value, the majority value for that feature is considered as the final value.

Data Alternation and Removal:

We removed feature 'fnlwgt' as it is not related to the people being studied and it has unrelated. Since, features 'education' and 'education-num' have the same meaning. Here, we only consider 'education-num'. Also, we convert 'hours-per-week' feature to a categorial feature which can take '<40', '40-60', and '>60'. Further, we apply get_dummies() function to convert categorical variable into dummy/indicator variables. Thus, in overall, our data has 108 features. At the end we normalized the data to have mean zero and variance one. Our training data includes %75 of the whole data set and the rest is the test data.

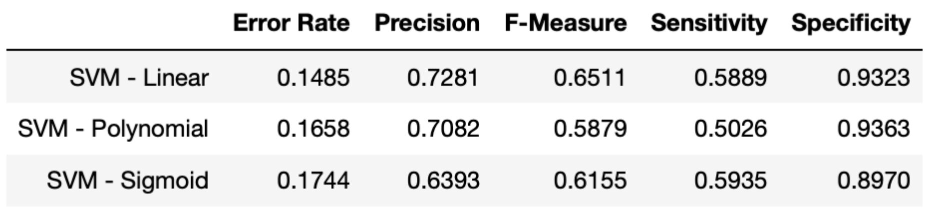
Evaluation:

Before designing different models we write a function called *evaluate_model()* to evaluate models based on actual and predicted outputs. Output of this function are error_rate, precision, f_measure, sensitivity and specificity. At the end, we are using these information to choose the best model for this data.

Model Design:

Support Vector Machine:

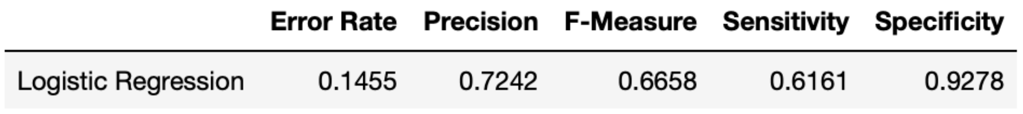
Since our data size is not very large we decide to use SVM model. We trained SVM models with three different kernel functions, linear, polynomial and sigmoid kernels. The results are as follow and SVM-Linear performs better.



	Error Rate	Precision	F-Measure	Sensitivity	Specificity
SVM - Linear	0.1485	0.7281	0.6511	0.5889	0.9323
SVM - Polynomial	0.1658	0.7082	0.5879	0.5026	0.9363
SVM - Sigmoid	0.1744	0.6393	0.6155	0.5935	0.8970

Logistic Regression:

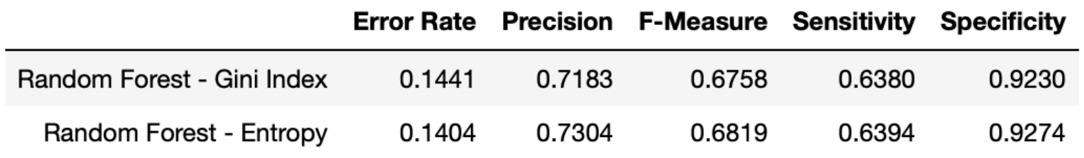
The next model which we experiment with is logistic regression. We choose this classifier due to its strength in dealing with Boolean values, due to our encoding of categorical features. For this model we use l2 regularization. The table blow summarized the results:



	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Logistic Regression	0.1455	0.7242	0.6658	0.6161	0.9278

Random forest:

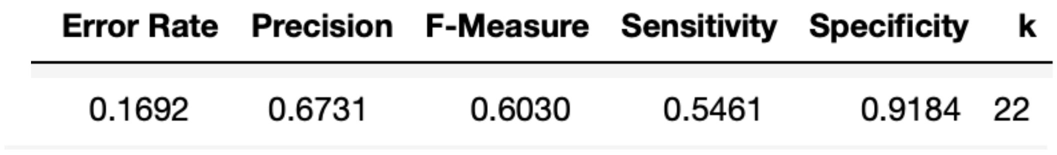
In this model we used both Gini Index and Entropy to decide the splits of the decision trees in the Random Forest. We experiment with 150 number of decision trees in the Random Forest. Also, we set minimum number of samples to split an internal node and minimum number of samples required at leaf node to be 2 and 1 respectively.



	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Random Forest - Gini Index	0.1441	0.7183	0.6758	0.6380	0.9230
Random Forest - Entropy	0.1404	0.7304	0.6819	0.6394	0.9274

K Nearest Neighbors:

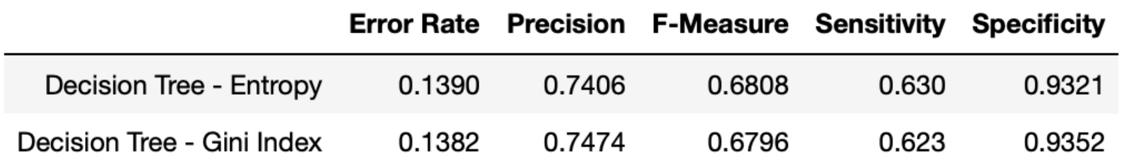
We evaluated the performance of K Nearest Neighbor classifiers for different values of K ranging from 1 to 50 in steps of 3. The best K with the minimum error rate is 22.



Error Rate	Precision	F-Measure	Sensitivity	Specificity	k
0.1692	0.6731	0.6030	0.5461	0.9184	22

Decision Tree:

To split a node in a decision tree, we use both Gini and Entropy. The best MinParent and min_sample_leaf are set 32 and 38. As can be seen below, the Gini index gives slightly a better error rate while the method using Entropy gives a higher F-measure.

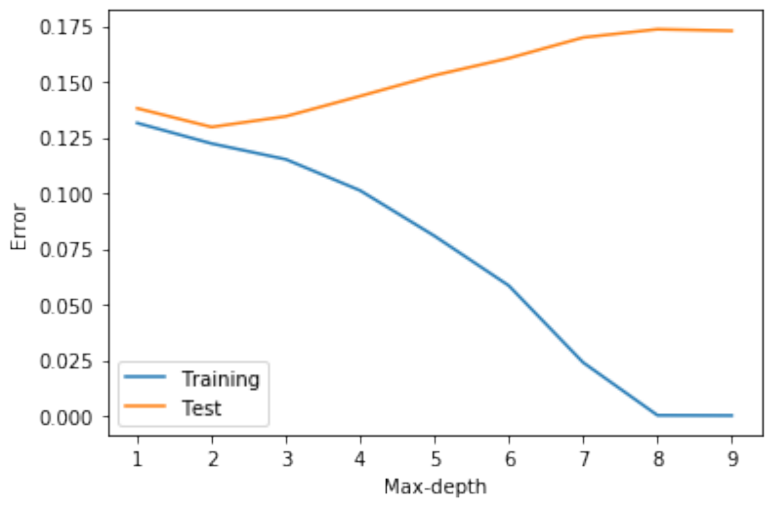


	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Decision Tree - Entropy	0.1390	0.7406	0.6808	0.630	0.9321
Decision Tree - Gini Index	0.1382	0.7474	0.6796	0.623	0.9352

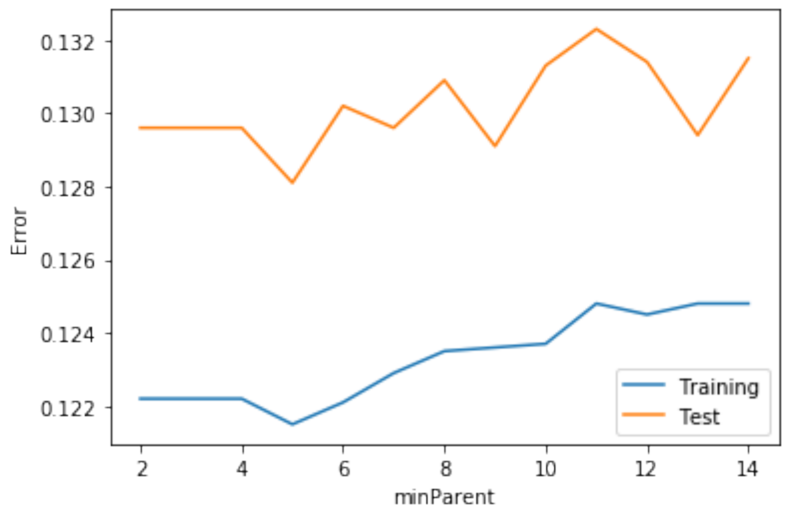
Further, the feature reduction using PCA does not seem to improve any measure.

Adaboost:

We used 100 decision tree classifiers as the base estimators of Adaboost. The following diagram shows that max-depth = 2 gives the minimum test error, while there is clear overfitting for max-depth > 2.



Now given the best max-depth=2, different setting of minParent is tested, in the form of 2^i for i in the range [2,15]:



Adaboost errors on various minParent settings

As can be seen, minParent = 2^5 gives the minimum error when max-depth = 2. The overall performance of Adaboost is given bellow:

	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Adaboost	0.1246	0.767	0.7186	0.6759	0.9368

Dimensionality Reduction with PCA on Adaboost

We project the data into a lower dimensional space, using PCA. It appears that number of component having variance over 0.9 is 80. Therefore, the first 81 components explain 90% of the variance. We set n_components = 81 and transform the training data using PCA. Learning on the new training data with Adaboost gives the following performance:

	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Adaboost-PCA	0.1583	0.6842	0.6437	0.6077	0.9137

As illustrated above, using PCA does not improve any measures.

Neural Network:

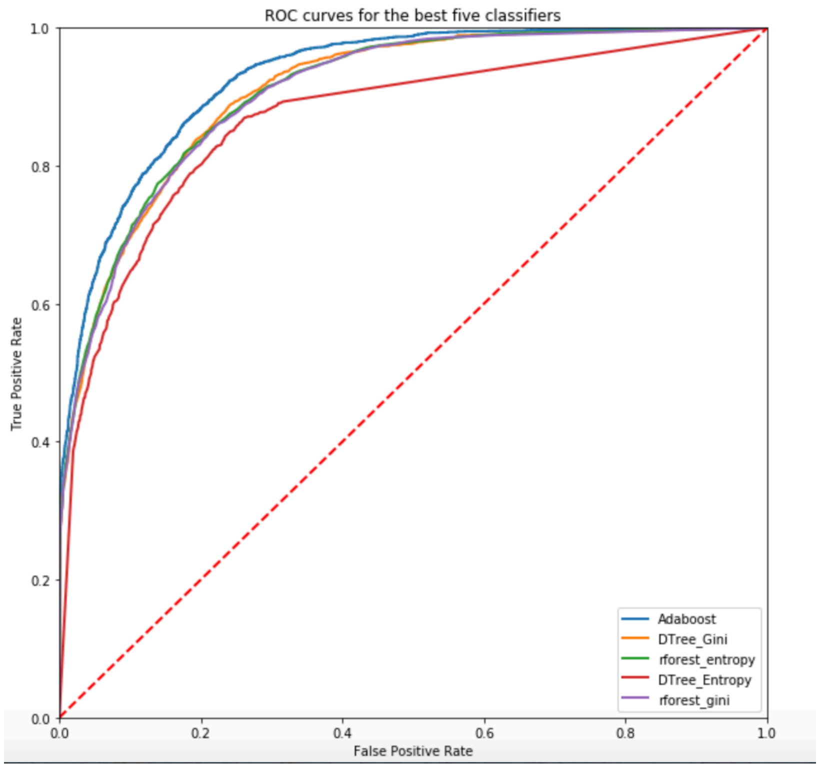
We experimented with various activation functions for an artificial neural network. And use the 'adam' solver for weight optimization as it is suited better for large data-sets. We also varied the number of hidden nodes in layers 1 and 2 for different activation functions.

	Error Rate	Precision	F-Measure	Sensitivity	Specificity
ANN - Tanh	0.1480	0.7122	0.6644	0.6227	0.9226
ANN - ReLU	0.1431	0.7335	0.6691	0.6150	0.9312
ANN - Logistic	0.1461	0.7155	0.6696	0.6293	0.9230
ANN - Identity	0.1452	0.7245	0.6669	0.6178	0.9277

Model Comparison:

By comparing output of evaluate_model() function for each trained classifier, we can see that the Adaboost model has the highest accuracy=0.8754 (1-error-rate). It is followed by the decision tree using Gini index (accuracy=0.8618).

We also plot the ROC curves of five models with the best values for accuracy. It can be seen that the Adaboost model has the best characteristics as it is closest to the top right corner where true positive rate = 1 and false positive rate = 0.



Conclusion:

Of all the models we trained, the Adaboost model (without using PCA) performed the best on the test data:

	Error Rate	Precision	F-Measure	Sensitivity	Specificity
Adaboost	0.1246	0.767	0.7186	0.6759	0.9368

Distribution of Work:

Each member of the team contributed significantly to the overall direction of the project and decisions that were made throughout. As a whole, the team put ideas together by discussing topics that came up in class and applying them to the project.

Zahra Askarzadeh: She was responsible for support vector machine, logistic regression, random forest, neural network. She also finalized the report draft.

Elham Havvaei: She was responsible for implementing K-nearest neighbors, Adaboost, Decision tree and dimensionality reduction(PCA). She also finalized the report draft.

Saumitra Prabhat Kadge: was responsible for developing the pipelines for preprocessing and building the project's notebook where we optimized hyper-parameters. He was also responsible for evaluating testing error of finalized parameters and implementing gradient boosting trees.

References:

1. https://archive.ics.uci.edu/ml/datasets/adult - Dataset Source and Documentation
2. https://matplotlib.org/contents.html - Matplotlib Documentation
3. https://docs.scipy.org/doc/ – Numpy Documentation
4. https://pandas.pydata.org/pandas-docs/stable/ – Pandas Documentation
5. https://www.scikit-learn.org – Scikit Learn Documentation
6. https://pbpython.com/pandas-crosstab.html
7. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
8. https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/