# Income Classification

Mathematics , Imperial College London

Imperial College London

## Introduction

The aim was to construct predictive models of whether a person has an income above 50K using binary classification of varying complexity. The data consists of 32561 entries sampled from a 2018 census with the age, work class, years in education, job status, race and sex of individuals provided. Other categories such as capital loss and gain were provided but we dropped these from the data as they had value 0 in most cases and could complicate the models unnecessarily. Fortunately, no null values were encountered. Hence, we constructed 10-feature models with income <=50K as output.

## Methodology and Classifiers

We first use StandardScaler in sklearn to standardize the data for fair comparison of features. We divide the data up into 5 blocks of roughly equal sizes, and use 4 blocks to train the data set and 1 for testing the fit of the models. In order to prevent a bias towards the training data which may result in over fitting, though this may be marginal as the data set is very large, we use a five-fold cross-validation technique, setting aside a different block for testing each time and averaging the performance metrics across these. An 80-20 train-test split is suitable as any smaller testing set will have greater variance, while the training set is large enough to avoid bias. For each model, we found the confusion matrix, precision, recall and the F1-score [1]:

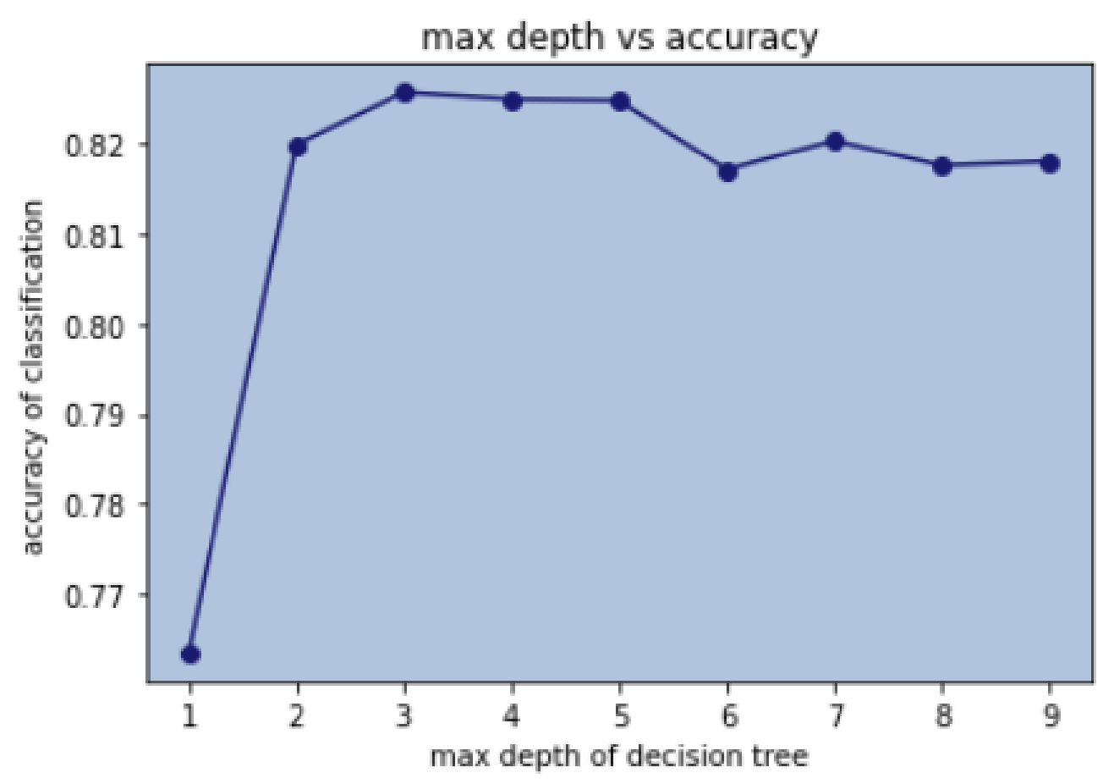
| | pos. prediction | neg. prediction |
|---|---|---|
| pos. class | TP | FN |
| neg. class | FP | TN |

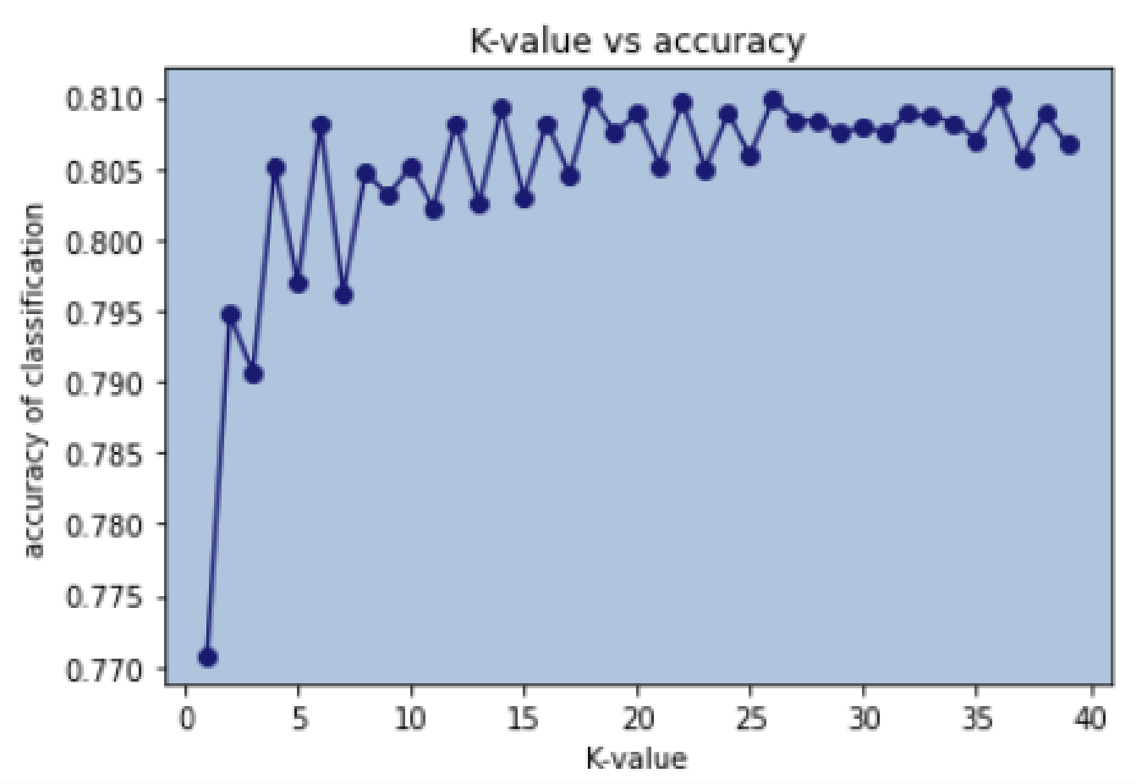
$$precision = \frac{TP}{TP+FP}, \qquad recall = \frac{TP}{TP+FN},$$

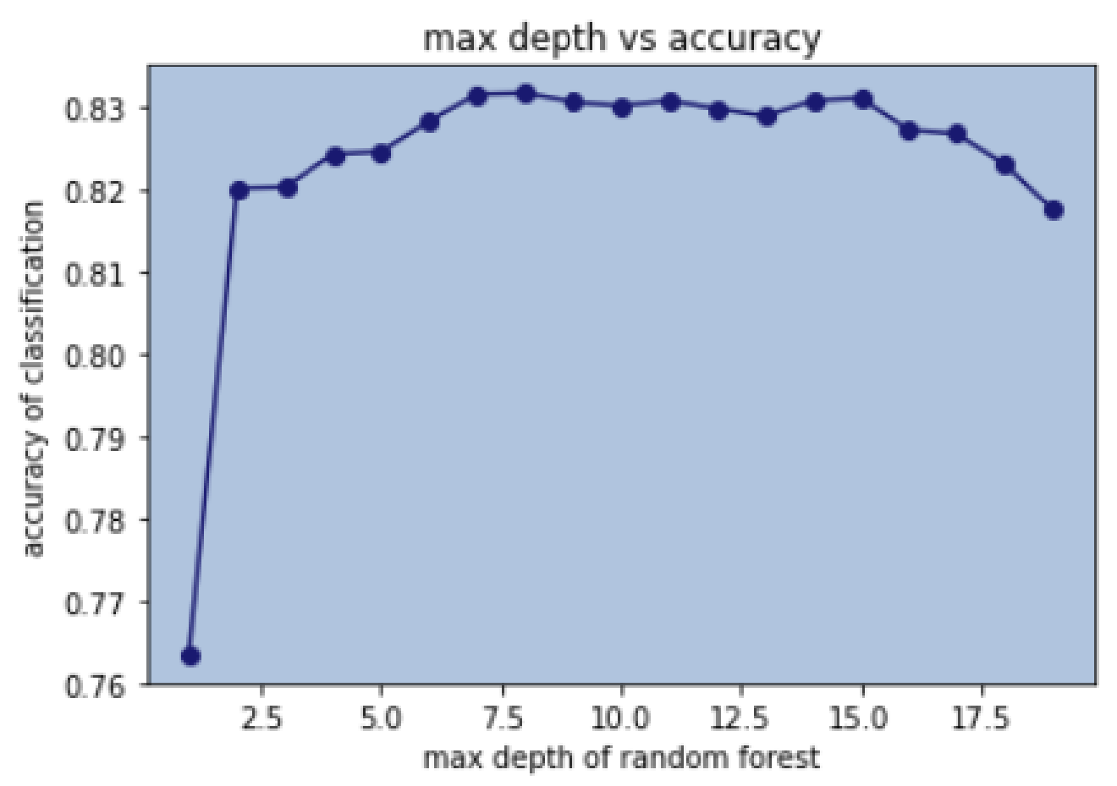
$$F1 - score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}},$$

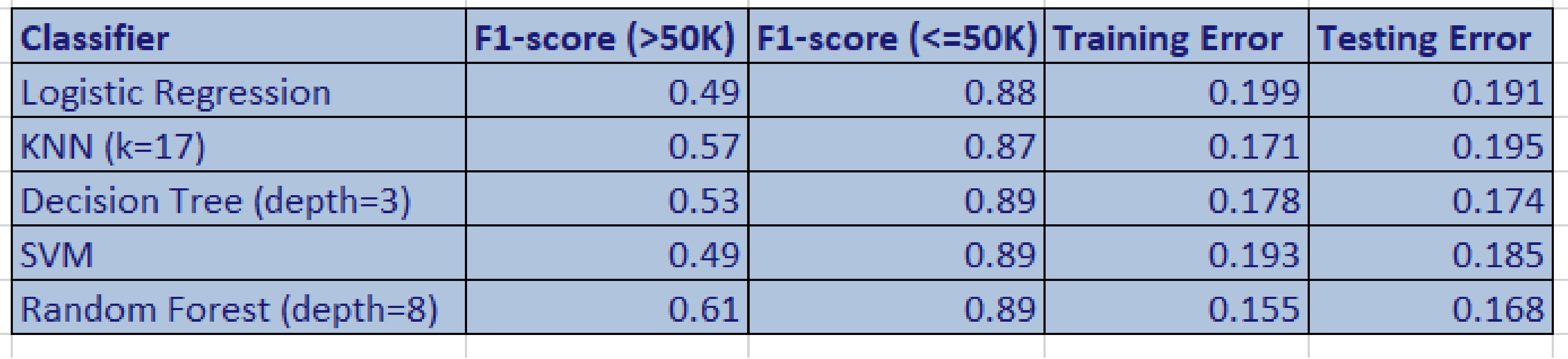
A pessimistic model, for example one that underestimates people's incomes, will have a high precision as it will infrequently wrongly classify an income as above 50K but a low recall. The F1-score takes the harmonic mean of precision and recall. Depending on the way the data is used, precision or recall may be more important - precision will be if for example anyone earning above 50K is to be taxed extra, such that a false positive is costly. The classification models range from a simple logistic regression to support vector machine and random forests.

## Graphs









| Classifier | F1-score (>50K) | F1-score (<=50K) | Training Error | Testing Error |
|---|---|---|---|---|
| Logistic Regression | 0.49 | 0.88 | 0.199 | 0.191 |
| KNN (k=17) | 0.57 | 0.87 | 0.171 | 0.195 |
| Decision Tree (depth=3) | 0.53 | 0.89 | 0.178 | 0.174 |
| SVM | 0.49 | 0.89 | 0.193 | 0.185 |
| Random Forest (depth=8) | 0.61 | 0.89 | 0.155 | 0.168 |

## Random Forest

The use of multiple trees in random forests in theory reduces over-fitting, though in practise our model shows slightly more bias than the decision tree. Random forest classification runs efficiently on large databases such as this one and is capable of estimating missing data (though this wasn't needed). A random forest constructs multiple decision trees and the decision of the majority is chosen. Parallel ensembling [2] is used to fit several classifiers in parallel on sub-samples and, followed by majority voting, this minimises over-fitting. The graph shows that beyond a depth of 8 the accuracy in fact decreases, so we set the max depth parameter to 8.

## Decision Tree

Decision trees work by splitting the data according to various features, finding the classification error from each split and minimising it. We specifically minimise entropy [3]. Entropy, H(x), is a measure of randomness in the data-set, while information gain is the decrease in entropy after splitting the tree.

$$H(x) = -\sum_{k=1}^{n} p(x_i) log(p(x_i))$$

With added depth, the training error drops to 0 but the model is prone to over-fitting. From the graph we can see that a depth of 3 provides a compromise - in fact, testing error is lower than training error. Another means of reducing over-fitting is by ensuring a minimum node size - we don't split if there are too few data points.

## K Nearest Neighbours

This model simply computes the Euclidean distance between each point and its neighbours, assigning it the majority class of its k nearest. An easier alternative is the Manhattan distance [4] , which simply sums the x and y coordinate differences. The accuracy increases up till about k=17 then oscillates as k is increased, hence 17 is the optimal value as anything beyond is unnecessary complexity and will increase running time.

## Logistic Regression

Logistic regression is used as a baseline. This model scales values between 0 and 1 on use of the sigmoid function:

$$sigmoid(score) = \frac{1}{1+exp(-score(x_i))}$$

Then gradient descent [5] is used to iterate towards the minimum, taking repeated steps in the opposite direction of the gradient of the function at the current point providing the optimal weights for the model.

## Support Vector Machine

A Support Vector Classifier calculates the margin as the shortest distance between a point and the threshold. The model determines a hyper-plane from the margin and categorises points above and below it. Hence, performance with SVC is poor in the case where multiple planes would provide a better split for the feature space. For example, both very young and very old age might suggest a low income, yet SVC is designed to make only a single cut off. However, SVM uses kernel functions to find SVC in higher dimensions, overcoming the issue. Kernel functions can be linear or polynomial but the best is the radial kernel which translates the relationship between points to infinite dimensions:

$$radial = exp(-\gamma(a-b)^2))$$

## Conclusions

All 5 models produce an F1-score of almost 0.9 for <=50K so this criterion can be disregarded, hence we compare F1-scores for >50K. As expected, logistic regression is lowest at 0.49. We can also eliminate SVM and Decision Tree from contention as they hardly improve on the score for logistic regression, if at all. Logistic, decision tree and SVM all have a test error lower than the training accuracy, while KNN and Random Forests both seem to over-fit. However, in the case of the latter, although the test error is worse than the train error by a small margin, both test and train error are lower than for the other models. Overall, we recommend using Random Forest to classify this data-set, despite minor over-fitting. An additional advantage is that tree-based methods are easier to interpret, which may help in understanding the most important factors driving wealth. Though it is computationally less efficient than logistic regression, for example, modelling and prediction time are almost instantaneous anyway.

## References

[1] R. Trifonov. Binary Classification Algorithms. International Journal of Development Research, 2017.

[2] L. Cheng. Basic Ensemble Learning. Towards Data Science, 2019.

[3] I. Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. Springer Nature, 2021.

[4] K. Gohrani. Different Types of Distance Metrics used in Machine Learning. Machine Learning enthusiast, 2019.

[5] D. Jurafsky. Logistic Regression. Stanford University, 2020.