Algorithms

2.1 For each of the five algorithms list key strength and key weakness. Use no more than 250 words in total.

1. Gaussian Naive Bayes

Strength - deals with the curse of dimensionality problem, reducing the need for a very large training set in case of many features. Moreover, it works very well with small datasets.

Weakness - slow, because for every new sample it has to calculate every class conditional probability for each of its features.

2. Decision Tree

Strength - useful in case we want to find a non-linear decision boundary for our regression or classification problem.

Weakness - decision trees are unstable. This means that a small change in the data can cause huge changes in the tree.

3. K-Nearest Neighbours

Strength - can be updated on the fly as new training data is added at little cost.

Weakness - slow and memory inefficient, because of many distance estimations for a new sample to predict and the complete training set has to be stored.

4. Logistic Regression

Strength - simple algorithm which can adapt very fast to new training samples as it uses gradient a(/de)scent.

Weakness - does not work good with small datasets, gradient accent needs many training samples to converge properly.

5. Support Vector Machine

Strength - memory efficient algorithm which works very well with linearly separable data.

Weakness - inefficient with large datasets, because the training time is slow and all the support vectors have to be stored.

2.2 Carefully read the Scikit-learn hyper-parameter documentation for each of the five algorithms. Based on this documentation explain how the previously mentioned hyper-parameters effect the algorithms and their performance. Express yourself clearly and provide your reasoning. Use no more than 300 words in total.

1. K-Nearest Neighbours:

"n_neighbours" - the bigger its value is, the more distances are taken into account for prediction, which leads to better performance against noisy data, but also to more unclear decision boundary and time complexity. If its value is too small, the algorithm can be badly affected by noise and incorrect feature scaling of the training data.

"weights" - can lead to bad performance when set to 'distance' and there is incorrect feature scaling.

2. DecisionTree Classifier:

"max_depth" and "min_samples_leaf" - determine how much the algorithm generalises from the data, the degree of smoothing and the degree of overfitting.

Small max_depth or large min_samples_leaf lead to faster training and prediction, but also can lead to underfitting.

If max_depth is large, then the algorithm can experience overfitting if min_samples_leaf is also too small.

3. SVC:

"C" - determines the trade-off the algorithm makes between making miss-classification as small as possible and finding the largest margin between the classes. Higher values lead to smaller margins and may lead to overfitting, while very small values lead to larger margins that in turn cause more misclassification and underfitting.

"kernel" - allows the classifier to be fitted to non-linearly separable data, thus adapting it to different kinds of separation between classes.

4. Logistic Regression:

"C" and "penalty" - control the regularisation applied in the algorithm to adjust the weights coefficients so that overfitting or underfitting does not happen. If penalty is not 'none', the lower the C, the more penalty is applied in case of large weight coefficients. The type of penalty norm usually does not make a huge difference on the performance.

"random_state" - controls the randomness of the train/test split in case we apply cross validation etc. In practice, more realistic algorithm performance it should not be fixed.

US Census

Data Exploration

1. Explore the features and target variables of the dataset. What is the right performance metric to use for this dataset? Clearly explain which performance metric you choose and why. Use no more than 125 words.

There are 11 features and 1 target variable - the salary. A good observation is that the target data is not evenly split: 12360 out of 16281 samples are negative. Given that, using solely accuracy, recall, precision or specificity will not give us a good metric for our model. The problem with it is that it equalises the misclassification. Usually the costs are not the same and having in mind it is not specified, it is better to go for AUC/ROC Curve. AUC/ROC Curve is really useful instrument which can give us good visualisation of both cases of misclassification. As long as we do not need exact probability of an entry belonging to target variable 0 or 1, we are good to go.

2. Algorithmic bias can be a real problem in Machine Learning. So based on this, should we use the Race and the Sex features in our machine learning algorithm? Clearly explain what you believe, also provide us with arguments why. Note this question will be graded based only on your argumentation. Use no more than 75 words.

Using gender and race can lead to really biased algorithm. There are already existing cases where people's CVs were discarded based only on the gender. These 2 features should not affect our algorithm as they should not play a role in the salary a person receives. We can use them for statistics, but not for decision-making. Given that we believe that gender and race should not be included.

Data Preparations

1. This dataset hasn't been cleaned, yet. Do this by finding all the missing values and handling them. How did you handle these missing values? Clearly explain which values were missing and how you handled them. Use no more than 100 words.

There are missing values in 4 columns: education-num(240), workclass(936), occupation(1181) and native-country(300).
'education-num': most likely user forgot to give input, we estimate (take the average) based on type of education. So if a person has type of education 'Masters', we set education-num as the average of all the people who did 'Masters'.
'workclass': empty because there is no occupation, below 50,000$ for sure.
'occupation': most likely the person is unemployed, below 50,000$ for sure.

'native-country': most likely user did not want to give input, we leave it like that.

2. All Scikit-learn's implementations of these algorithms expect numerical features. Check for all features if they are in numerical format. If not, transform these features into numerical ones. Clearly explain which features you transformed, how you transformed them and why these transformations. Use no more than 75 words.

The features that were transformed are 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'native-country'. All the unique values for each parameter are extracted and a dictionary is created with a structure like {'unique_value_1': 0}. Then the values are replaced with the corresponding index. This way the solution is scalable.

3. Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did to be systematic, what you did to get fair results, what trade-off accuracy vs resources trade-off, etc. Use no more than 200 words.

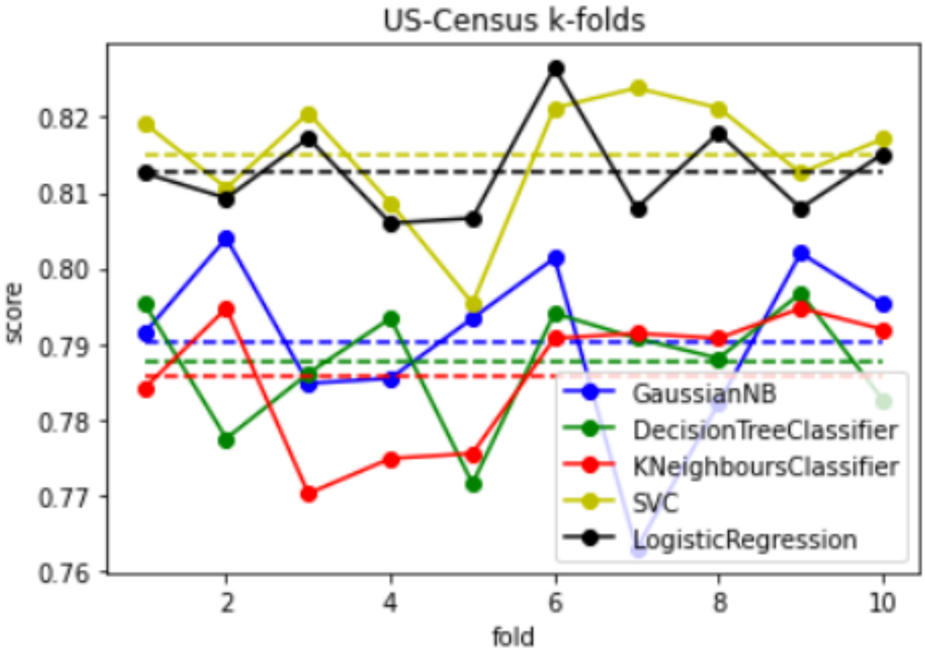
As discussed in the data exploration step - dropping 'sex' and 'race' as features should be done, as they should not influence the salary of a person.

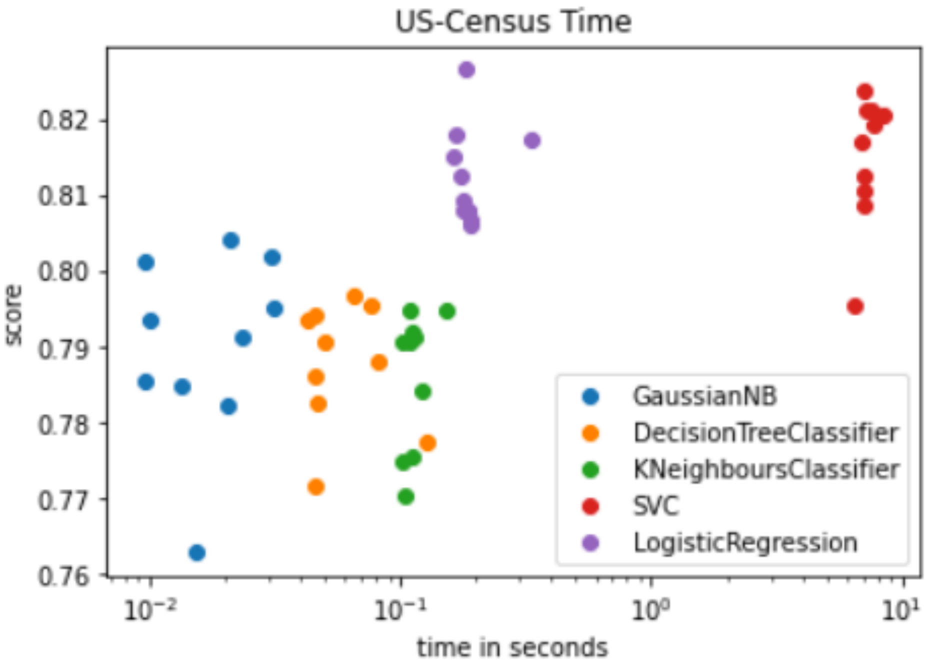
Experiments

1. Now set up your experiment. Clearly explain how you divided the data and how you ensured that your measurements are valid. Use no more than 100 words.

We will use cross-validate to verify the score of each algorithm. We will go through all of them and plot the algorithms' score and time they take as well, in order to pick the one which suits our needs. Moreover, we will use Excel table to keep track of the hyper-parameters tuning and check how the different adjustments influence the outcome of the algorithm. This will help us for not running same experiments twice and have a clearer vision of the direction in which we go.

2. Fit the five algorithms using the default hyper-parameters from section 2.1.Create a useful plot that shows the performances of the algorithms. Clearly explain what this plot tells us about the performances of the algorithms. Also, clearly explain why you think some algorithms perform better than others. Use no more than 150 words and two plots (but 1 is sufficient).



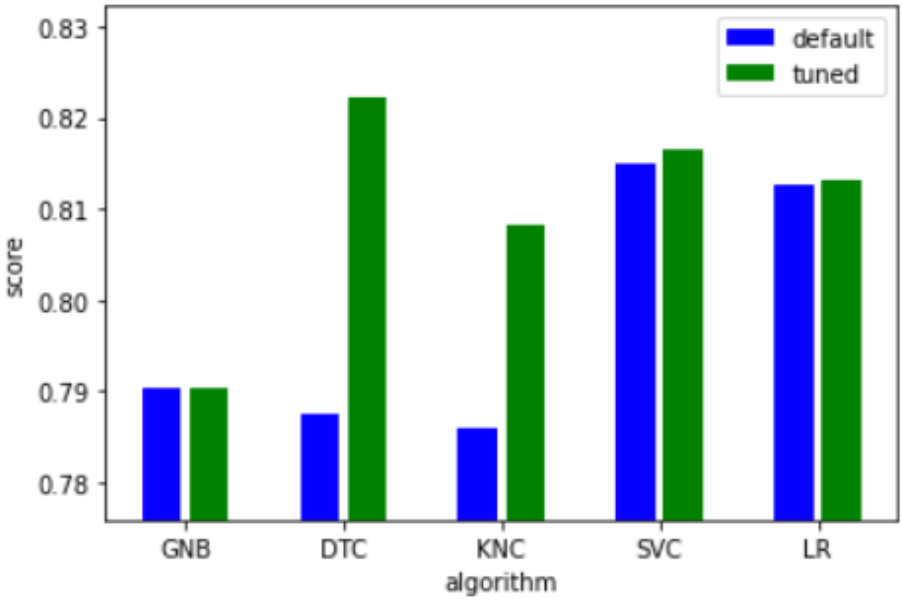


There are 2 plots shown. On the first one we can observe the fold number on the x-axis and test score on the y-axis. For each algorithm we can see its score for each fold and a mean line as well. Observing exact score for fold doesn't matter that much, what matters more is to see the bias that the algorithm has compared to the mean. In the second plot we can observe the time (fit + score) for each fold and choose the algorithm suits us in terms of score/time. We can see that SVC and Logistic Regression have the highest score, however, SVC is much slower. This is most probably because those 2 algorithms are specifically designed for binary classification.

3. Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did to be systematic, what you did to get fair results, what trade-off accuracy vs resources trade-off, etc. Use no more than 200 words.

We have used **GridSearchCV** and first we put 3 random values per parameter for each algorithm and then adjusted them for lower/higher ones depending if the outcome was the min/max until we found the optimal hyperparameters. We have tried to test different edge values as well (for example C=0.00001 and C=200) in order to check if we are not in a local maximum.
Logistic Regression: best penalty and C parameters are 2 and 'l2' respectively, with score: 0.813100123061669. Penalty and C are related with overfitting and regulation of the parameters.
SVC: best kernel and C parameters are 'rbf' and 100 respectively, with score: 0.8173387310828331. The default parameter 'rbf' creates a good enough hyperplane. While the correction of C has to be higher.
Decision Tree Classifier: best max_depth and min_samples_leaf are 11 and 20 respectively, with score: 0.8255513535686788. Not having 'max_depth' results in worse result, as the tree was overfitting before, thus pruning is needed. 'min_samples_leaf' is a 20, which increases the confidence when reaching leaf.
K-Neighbors Classifier: best n_neighbors and weights are 24 and 'uniform' respectively, with score: 0.8083318672688865.

4. Compare the performance of the algorithms with and without hyper-parameter tuning. How did the tuning affect your result? Clearly explain the results and the differences. Use no more than 100 words and two plots (but 1 is sufficient).



The plot shows the average score of 10 folds for each algorithm before and after being tuned. We can clearly see that the Decision Tree Classifier and the K-Nearest Neighbours Classifier benefited the most from the tuning. SVC and Linear Regression got better score but just by slightly bit, which is negligible. Gaussian Naive Bayes is the same, as no tuning was done on it. After the tuning DTC is clearly the best algorithm to choose, not only that, but from previous tests we know that the time it takes is also one of the best.

5. Select your best algorithm for this dataset and use it to make your predictions for the unknown samples. Please note in you algorithm which algorithm you chose.

DecisionTreeClassifier(max_depth=11, min_samples_leaf=20)

MNIST

Data Exploration

1. Explore the dataset by plotting the same image from both datasets side by side. How do these images compare? Which dataset do you expect to perform better?Clearly explain why you suspect that. Use no more than 75 words.

First images from the datasets are 1s. We can see the two images differ in the amount of information they provide because of the difference in dimensionality.
The digits in the dataset are not uniformly distributed, because 0 is predominant, while 5 and 7 are least frequent.
Because the data in the second dataset contains more correlation and more redundant information the resulting presumption is that the smaller dataset will on average perform better.

Data Preparations

1. Examine the features of both the datasets and decide if you need to do any data cleaning or preprocessing. If not, clearly explain why not. If yes, clearly explain why and what you did. Use no more than 100 words.

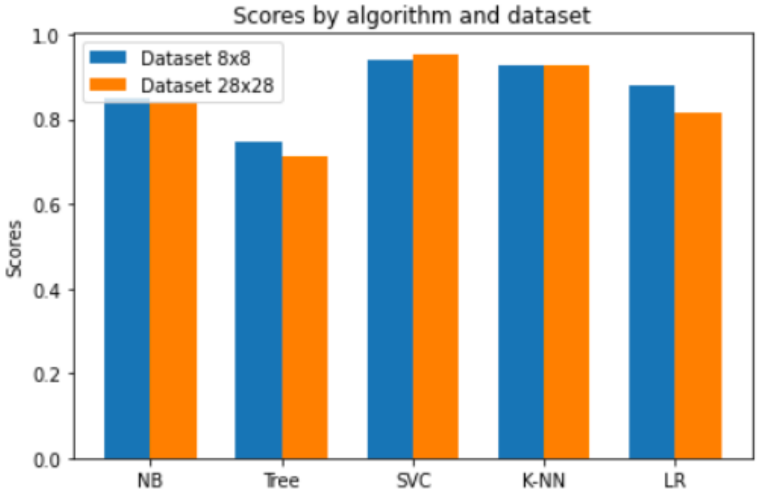
Because we want every sample to be a feature vector in order to work with the classifiers, we will reshape the data in both datasets.
No data cleaning is needed, because the dataset does not contain missing or null values. However, there is lots of redundant data and to get rid of it, we will perform PCA dimensionality reduction. In this way, we also significantly decrease the training time of algorithms like SVM and Decision Tree.
The data has feature scaling and so there is little need to normalise the data for K-Nearest Neighbours, Logistic Regression or SVM.

Experiments

1. Now set up your experiment. Clearly explain how you divided the data and how you ensured a valid measurement. Use no more than 100 words.

We will divide the train set into two, leaving 10% of the data as out validation set. We will also use a standard 10-fold cross-validation to get an average estimate of our classifiers' performance. In this way, we also ensure that the whole dataset will be used for testing, leading to more correct classifier evaluation.
However, we will not do it now because Scikit-learn already provides a method for performing cross-validation on a given dataset and labels.

2. Fit the five algorithms using Scikit-learn's default hyper-parameters. Create a useful plot that shows the performances of the algorithms. Clearly explain what these plots tell us about the performances of the algorithms. Also, clearly explain why you think some algorithms perform better than others and why some of them preform better on one dataset than the other. Use no more than 200 words and two plots (but 1 is sufficient).



The bar plot below shows the average accuracy of the five algorithms cross-validated using 90% train data and 10% validation data. As we can see, the performance is from worse to best:
Decision Tree < Naive Bayes < Logistic Regression < K-Nearest Neighbours < SVC

The plot also shows that on average:

- SVC and K-NN perform better with the larger dataset
- GaussianNB, Decision Tree and Logistic Regression perform better with the compressed dataset.

Explanation of results is as follows:

1) Decision tree does not perform well because classes are not uniformly distributed.
2) The Naive Bayes also does not perform well as its assumtion of feature independence is incorrect.
3) Even after PCA, there is noise in the data, leading to lack of good convergence for gradient ascent of Logistic Regression.
4) K-NN and SVC usually perform well with large datasets and in this case weighted K-NN considers the distances to neighbours as well, making it less affected by the non-uniform digit distribution.

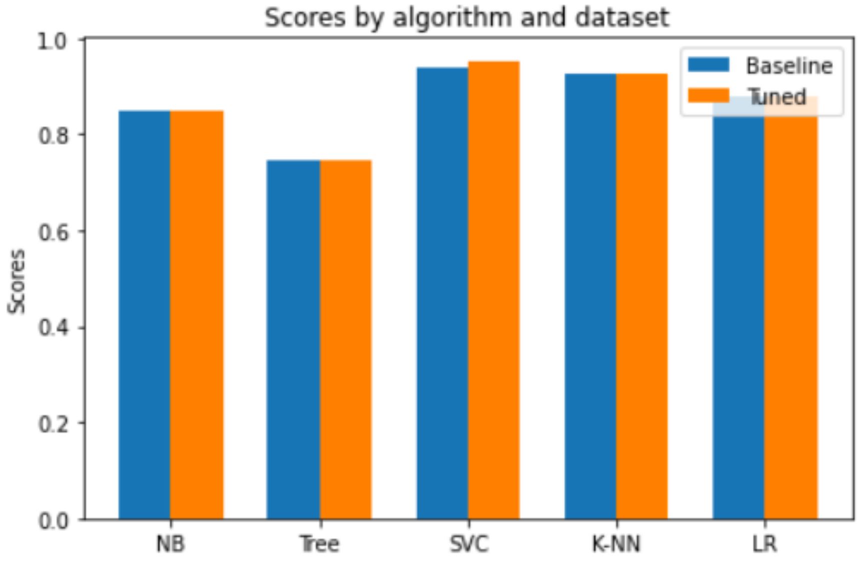
3. Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did and how you did this. Use no more than 200 words.

To perform hyper-parameter tuning, we use double cross validation. To do this, we use a StratifiedKFold object to make 6 splits of our training MNIST dataset into training and validation set. We use a GridSearchCV object to perform the inner cross validation. For every model and every split of the data, we fit the split's train data into the GridSearchCV object, which automatically performs the inner cross validation for every combination of the provided values for the model's hyper-parameters. The best performing combinations for every split are then compared in the outer CV with the split's validation set.

The best performing combination in the outer CV is used to tune the model.

We perform this for both the large and the compressed MNIST training datasets.

4. Compare the performance of the algorithms with and without hyper-parameter tuning. Also, make a comparison with your original baseline. How did the tuning affect your result? Clearly explain the results and the differences. Use no more than 200 words and two plots (but 1 is sufficient).



After we tuned our models, we will compare their performance with that of the baseline versions by again performing a 10-fold cross validation. On the plot below we can see the results of testing our algorithms on the smaller PCA-reduced dataset with and without hyper-parameter tuning.

Clearly only the K-NN and SVC classifiers managed to be improved in the process. For SVC, this was done by just decreasing the regularisation parameter, thus reducing the

amount of overfitting. For K-Nearest Neighbours, we again use the weighted k-nn version of the algorithm, but now with a smaller number of neighbours to use for prediction.
For the other 2 tuned algorithms - Logistic Regression and Decision Tree, it could be that the method or comparison metric for tuning was not efficient enough, leading to a badly chosen combination of parameters.

5. Compare the performance of the algorithms with the 8x8 and 28x28 features.What effect do the additional features have? Also, state what you think causes this effect. Use no more than 75 words.

The results by now showed 3 of the 5 algorithms perform worse with the larger dataset, which corresponds with the initial assumption.

Presumably, Decision Tree and Logistic Regression are affected by the general curse of dimensionality problem, while Naive Bayes is affected by the increase of feature correlation and noise in the data.

SVC and K-NN are almost not affected by the number of features.

6. Select your best algorithm for this dataset and use it to make your predictions for the unknown samples. Feel free to use either the 8x8 or 28x28 features. Please note in your report which algorithm and feature set you chose.

SVC(C=10)

Conclusion

1. Which conclusions can we draw about the five algorithms examined during this assignment? For each algorithm briefly discuss the key thing you noticed about it during this assignment. Use no more than 250 words in total.

1) **Gaussian Naive Bayes**: MNIST dataset showed that less features make GNB much more accurate.
   However, in US-Census it performed worse than the other algorithms after they have been tuned, otherwise it was quite similar to some of them.
   What it excels in is the speed.

2) **Decision Tree Classifier**: If the features have bigger correlation, the algorithm is not affected that much by the change of dimensionality.
   From the US-Census we have observed that this classifier is the one with the highest score compared to the others after hyper-parameter tuning, which means it is a good classifier for binary classification.

3) **Support Vector Classifier**: SVC has the best performance before the hyper-parameter tuning. However, the actual tuning didn’t affect it that much in both datasets and in the US-Census the DTC came above it. The biggest drawback of the SVC is the time it takes - fit + score time for US-Census was ~6 seconds compared to all of the others which were ~0.1 seconds.

4) **Linear Regression**: In MNIST, LRC was greatly affected by the number of parameters. In both datasets LRC's score was barely improved after the hyper-parameter tuning. In the US-Census it scored almost the same as SVC, which was the highest before the tuning.

5) **K-Nearest Neighbours**: After comparing the performance of KNC's performance in the 2 datasets, we concluded that it performs better with numerical data than categorical one. It had the worst score (before tuning) in US-Census and second best in the MNIST.