Report

# Clustering

**Report Task 1**

The K-means algorithm separates data by minimizing the sum of squared distances between each data point. The Euclidean distances between data points tend to become uniformly large when dealing with high-dimensional data [1]. To tackle this problem, I reduced the dimensionality of the data using PCA with the “mle” parameter which determines the most optimal number of principal components. Additionally, I scaled the data to have unit variance because the kmeans defines cluster with equal variance across all dimensions. I avoided subtracting the mean as this approach tended to create 1 big cluster. Then I apply KMeans on the scaled data with initialization – “init” parameter set to “random” which tends to produce a more even size of clusters and set the number of initializations counts to 10 for better initial centroids.

I applied GaussianMixture with the same preprocessed data from Kmeans since they both rely on distance metrics. I set the clusters to have the same covariance matrix because different covariance matrixes have a lot of parameters which can lead to overfitting(recognising irrelevant variances) given the high dimensionality of the data.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **K-means** | **Gaussian mixture model** | **Random baseline** |
| **Error count** | 3712128 | 2211682 | 6183659 |
| **Total pairs with the same label** | 7190302 | 7190302 | 7190302 |
| **Accuracy percentage** | 51.62% | 69.25% | 14.29% |

**Report Task 2**

**Report Task 3**

K-means struggles with high dimensional data (curse of dimensionality). In 50 dimensions Euclidean distances became less meaningful and the data is unlikely spherical and of equal variance as Kmeans assumes. The Gaussian Mixture model gives better results due to its ability to capture variations in the data that K-means isn’t able to but may be limited in its accuracy due to the structure of the data and initial centroids. The random baseline has a very low accuracy which highlights that the k-means and GMM models are capturing meaningful patterns and are far better than random guessing.

# Classification

**Report Task 4**

Support vector machines require extensive training time for large datasets like the Covertype dataset. For non-linear SVMs usually the Gram matrix is computed making the training time scale at least quadratically with the number of samples which becomes impractical beyond the thousands of data points [2]. The LinearSVC is a faster implementation of a Support Vector Machine in the case of a linear kernel. It uses liblinear algorithm which avoids computing the Gram matrix and scales linearly to the number of samples. The disadvantage of this model is the inability to capture non-linear patterns within the data and achieves an accuracy of around 70%.

**Report Task 5**

|  |  |  |
| --- | --- | --- |
| **lbfgs** | 61.23 | NA |
| **liblinear** | 70.48 | 68.01 |
| **newton-cg** | 69.06 | NA |
| **newton-cholesky** | 71.52 | NA |
| **sag** | 67.86 | NA |
| **saga** | 68.40 | 67.60 |
|  | **L2** | **L1** |

The table below shows the results of running logistic regression with different solver and different regularization degrees. Seeing that the newton-cholesky solver gives the best results I cross-validated different regularization values 0.001, 0.01, 1, 10, and 100 for this solver, but they gave nearly identical results. Because on this solver the multinomial approach can’t be applied, I tried newton-cg, the best solver that allows this feature but got worse results of around 70% accuracy. My final model uses newton-cg with a regularization value C=1 with 50 iterations as convergence does happen in those 50.

As a decision tree requires little data preparation, so no scaling or dimension reduction is applied to the data. Initially, I ran the DecisionTreeClassifier with its default parameters without any constraints on the size of the tree with gini as the function for measuring the quality of the split. The accuracy score on the same training set was 100% which clearly stated that the model is overfitting the training data. Applying PCA to decrease the dimension with low variance gave the same result. Using cross validation I tried multiple maximum depth values shown on Figure 1. The graph shows that the accuracy doesn’t increase substantially for trees bigger than 25 layers. The same approach was used for the criterion (the function that measures the quality of the split when training the model) but all models showed similar results. That is why my model has parameters max depth of 25 and gives 98% accuracy on the training data which is a good balance between model complexity and accuracy.

Figure 1. The graph illustrates the relationship between the maximum depth of the decision tree and the accuracy of the model on the validation dataset.

For the ensemble method, I chose the random forest classifier. Running a single decision tree overfits the data and bagging models are used to reduce the variance while boosting is used more for reducing the bias [3]. To tune the parameters, I used gridSearch initially on 'n\_estimators': [50, 100], 'max\_depth': [10, 20, None], 'max\_features': ['sqrt', 'log2'] which showed that 20 is too low for max depth the number of estimators didn’t substantially change the results and the greater number of features is better. I fixed the number of estimators to 50. Using grid search again showed that 35 is a good max\_depth parameter so the next parameter to test was the max features but the results plateaued at 10. In conclusion, more complex models with bigger depth and higher number of features performed better but not for a big amount.

**Report Task 6**

Logistic regression is a linear model and is unable to capture the nonlinear relationships between features and underfits the data. These complex patterns can be captured by the decision tree leading to a significantly higher accuracy. The accuracy of the tree is further improved by the random forest by reducing overfitting and improving generalization. The relatively low improvement can be because the estimators of the forest are outputting similar results struggling to capture different aspects of the data. Additionally, the single tree already performs well, meaning the remaining errors are likely concentrated in dense regions of the data where classes are harder to distinguish.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Logistic Regression** | **Decision Trees** | **Ensemble method** |
| **Accuracy percentage** | 71.55% | 93.04% | 95.80% |

# Regression

**Report Task 7**

When we plot the data, we can notice that the data has some polynomial features. The 2 curves indicate that the function should be of degree 3, but I generated polynomial features from degrees 1 to 5 and validated them using cross-validation. The model with polynomial features of degree 3 showed the best results with the lowest mean squared error as shown on Figure 2.

A graph with a line

Description automatically generatedFor training the neural network, I chose the lbfgs optimizer considering the small size and dimensionality of the data [3]. I tried different architectures of the neural network on 2 different kinds of preprocessing of the data. The errors of the initial model with random architecture with 100 iterations and 0.01 learning rate is shown in Figure 3. This indicates that the learning rate is too big achieving steep improvements but can crash. So I lowered the learning rate to 0.0001 and increased the number of epochs to 500 for a more accurate and robust model. The first one was given the data as it is and the mean squared results from the cross-validation with a fold of 10. Considering that the dataset was scaled to a small range I assumed that it reduces the activation of non-linear transformations in the neural network, making it difficult for the model to recognise the polynomial behaviour. That is why I pre-processed the data for an input x to be [1, x, x^2, x^3]. The results from Figure 4 show that the model trained on the preprocessed data on average is superior to the other one. The best result showed that the NN with two hidden layers with 100 nodes and 10 nodes each, even though other architectures can be considered due to the high variable results.

Figure 5. The chart shows the mean squared errors on different architecture types of a neural network on processed and unprocessed data. (100,3) means 2 hidden layers, one with 100 and the other with 3 nodes

Figure 8. Training error over 100 iterations with a learning rate of 0.01



Figure 2. Mean Square Error vs Polynomial features degree for linear regression

first fold of 4 KFold

Assuming that the data is a polynomial function of degree 3 due to the 2 curves we will work on a y = w0 + w1\*x + w2\*x^2 + w3\*x3 with prior probabilities of N(0, 20) for w0, w1, w2 and N(0,10) for w3 as having more confidence in that the function is of degree 3 and a uniform distribution for the sigma. The figure shows that the prediction of the training data fits the data well.

**Report Task 8**

A graph with a red line

Description automatically generatedA graph with green dots and red lines

Description automatically generated

Figure 7. Test data (blue dots), linear regression predictions (red line))



Figure 3. Training data (green dots), linear regression predictions (red line)



A graph with green and red dots

Description automatically generated

A graph with a red line

Description automatically generated

Figure 4. Test data (blue dots), linear regression predictions (red line)



Figure 6. Training data (green dots), neural network predictions (red line)



A graph with green and red dots

Description automatically generated

A graph with a red line

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | **Linear Regression** | **Neural Network** |
| **Mean squared error** | 75.07 | 160492.01 |

Since the input data is one-dimensional, the Linear Regression can capture the relatively simple relationship. Even though the Neural Network didn’t give bad results, this model is designed to handle complex, high-dimensional patterns because it requires more data to generalize properly. For this type of data, the simplicity and the deterministic nature of the linear regression model make it superior.

# Hidden Markov Model