CM3015 Mid-term coursework

Abstract

This project sought to identify the most effective machine learning model for the Iris flower dataset through a comparative analysis of two algorithms: k-Nearest Neighbours (kNN) and Decision Trees. Optimal parameters for each model were determined through tuning, and their performance was evaluated using key classification metrics. The results revealed that both kNN and Decision Trees reflected similar accuracy scores, highlighting the significance of further analysis and testing data to evaluate this aim further.

Introduction

2.1 Exploratory Data Analysis - Iris Dataset:

- The data set contains 3 classes with 50 instances each, and 150 instances in total, where each class refers to a type of iris plant (target).
  - **Class** : setosa, versicolour, virginica (represented by target values 0-2)
  - **Features**: sepal length, sepal width, petal length, petal width (cm)
    - We will be training our models based on these parameters and further use them to predict the flower classes.

Strategy and Evaluation Metrics based on data sample

- Evaluation Metric: **Accuracy**
  - **Balance:** as seen in 2.2.1 (Summary Statistics), the dataset has an equal number of instances for each class (50 per class)
    - Accuracy is calculated as the ratio of correctly predicted instances to the total number of instances.
    - In a balanced dataset, accuracy can be a reliable metric for assessing the overall correctness of predictions as it does not need to account for imbalances between classes or any trade offs in predicting a certain outcome such as false positives/negatives
- Cross-validation strategy: **k-fold cross-validation**
  - **Small dataset (<1000)**: seeing that dataset has a size of 150 instances, it is relatively small.
    - k-fold cross-validation involves dividing the dataset into 'k' folds, training the model on 'k-1' folds, and validating on the remaining fold. This process is repeated 'k' times, and the performance metrics are averaged.
    - For a small dataset this would be a reasonable strategy to assess the performance of the machine learning models as opposed to nested cross-validation, as splitting data into training and validation sets reduces the amount of data available for training, which may lead to overfitting or result in less robust models.

2.3.1 Feature relationships and distributions

a. Pair plots

- Shows scatter plots for each pair of features, showing how data points are distributed in two-dimensional space for each pair combination. Also combines colour mapping to represent the different classes.
  - Helps to identify patterns, trends and potential correlations between features based on classes.
- Also shows histograms for each feature along the diagonal of the pair plot
  - Displays distribution of values for that specific feature

i. ***Analysis of plot***

- Data is not normalised and features are not scaled by referring to the following:
    - Scale of Axes are dissimilar across different features
    - Distribution Shapes of histograms on the diagonal of the pair plot are varied
    - Presence of outliers

ii. ***With normalisation, we achieve the following:***

- Better Visualisation of Relationships:
    - Relationships between features are made more apparent with consistent scales
    - Facilitates model convergence such that machine learning algorithms converge faster and perform better
    - Mitigate impact of outliers as features with large scales might dominate the visualisation and skew the interpretation.
    - Normalised data is more interpretable, making it easier to compare the importance of different features.

2.3.2 Dataset standardisation

- Features are transformed using the z-score: $z = \frac{x - \mu}{\sigma} z = \frac{x - \mu}{\sigma}$, where $\mu\mu$ and $\sigma\sigma$ are the mean and standard deviation of xx, respectively.
    - The transformed features should centre around 0 and their standard deviations are 1.

a. Feature scaling function

- I have implemented a scaler function here to standardise the data, mimicking the StandardScaler from sklearn.preprocessing

b. Study scaled features to confirm their accuracy

- Checking data to confirm that the standardised data reflects that mean values are close to 0 and standard deviation values are close to 1.

c. Visualise the data after standardisation

- Pair plots are reproduced with the scaled data

i. ***Analysis of plot***

- Data is normalised, as indicated by the scale of axes and histograms that represent a normal distribution
- Separation of classes is also scaled as per distribution and visibly clearer to interpret

d. Boxplots

i. ***Analysis of plot***

- Box-plots show generally acceptable distributions, with only the feature “sepal width” having 3 outliers.
- The outliers are not too skewed from the data, and considering the small size od the dataset, it may be more prudent to retain these outliers for a more representative classification model

2.3 Establishing baseline (Naive Model)

- The baseline accuracy, often referred to as the "naive accuracy" or "majority class accuracy," is the accuracy that a very simple model, known as a naive or baseline model, would achieve. This simple model predicts the most frequent class for every instance in the dataset. The baseline accuracy is a useful reference point to assess the performance of more complex models.

- Baseline Accuracy = Number of instances in the majority class/Total number of instances
    - In this case, we count the instances of each class (50 each), and divide that by the sum of instances, being 50x3 (since we have 3 classes)
    - This gives us the Baseline accuracy of 50/150 = 0.33 approximately.

- This value sets a baseline to corroborate how well the ML model performs in this context, instead of using an arbitrary value or having to perform guesswork to evaluate the model.

Background

3.1 Classification algorithms

3.1.1 k-Nearest Neighbours (kNN):

- kNN is a simple and intuitive algorithm for classification. It predicts the class of a data point based on the majority class among its k nearest neighbours.
- Given a new data point, kNN identifies the k training data points that are closest to it (using a distance metric like Euclidean distance and an input value k).
- With the Iris dataset, given a new set of sepal length and sepal width (or any other combination of features), kNN calculates the distances to the training instances and assigns the most common class among the k nearest neighbours. The choice of k is crucial and can be determined through cross-validation.
- For example, given the value k = 3, and training data with the features sepal length and width, the algorithm calculates the distances between an instance and its neighbouring data points (training data), takes the majority of the 3 neighbours closest to the instance, and classifies that instance as the majority class identified.
- The optimal hyperparameters (e.g. distance metric (i.e. Euclidean/ Manhattan/ Cosine) and value of k) for the best classification model will be determined through iterating various combinations of these hyperparameters and identifying them based on the best scores produced by evaluation metrics those combinations provide.
- The optimal distance metric is dependant on the type of data, and the optimal value of k is when there is balance between reliability (data is classified fairly well), and generalisation to unseen data (algorithm is not too tailored to training data making it work well specifically only for only that set of data)

3.1.2 Decision Tree Classifier:

- Decision trees are a type of supervised learning algorithm used for classification.They recursively split the dataset into subsets based on the features, creating a tree-like structure.
- The algorithm selects the best feature to split the data at each node based on criteria such as Gini impurity or entropy, for classification. This process continues until a stopping condition is met, such as reaching a maximum depth or having a node with only one class.
- For the iris dataset, the decision tree algorithm recursively splits the dataset based on feature values to create a tree. For instance, it may find that a certain petal length threshold is effective in distinguishing between species.
- In simpler terms, the algorithm splits the data based on some criteria, evaluates the level of classification within the subsets using metrics that calculate how homogenous the data is, and continues this process within each subset until an adequate classification is achieved, or a certain parameter (set within the function) such as the maximum depth has been reached.
- Cross-validation helps ensure that the decision tree generalises well to unseen data, and just as in kNN, grid search can be used to optimise hyperparameters, such as the maximum depth of the tree.

3.2 Hyperparameter Tuning algorithms

3.2.1 k-Fold Cross-Validation:

- Cross-validation is a technique used to assess the performance of a machine learning model. k-Fold Cross-Validation involves dividing the dataset into k subsets (folds), using k-1 folds for training and the remaining fold for validation. This process is repeated k times, with each fold serving as the validation set exactly once.

- The model is trained and evaluated k times, and the performance metrics are averaged across the folds. This helps provide a more robust estimate of the model's performance and reduces the risk of overfitting or underfitting.

3.2.2 k-Fold Cross-Validation with Accuracy:
- Given the small size of the Iris dataset, k-fold cross-validation helps provide a more robust estimate of model performance by repeatedly splitting the dataset into training and validation sets. This is so that the limited data can be recycled per fold for training and testing data, to provide sufficient variability in data for the model to generalise itself with unseen data, making use of the limited resources prudently.
- K-fold cross-validation can be used to train and evaluate the models multiple times, ensuring that each instance serves both as part of the training and validation sets. The average accuracy across folds provides a more reliable estimate of how well the model is likely to perform on new, unseen data.

3.2.3 Grid Search:
- Grid search is a hyperparameter tuning technique used to find the optimal set of hyperparameters for a machine learning model. It involves defining a grid of hyperparameter values and systematically searching through the grid to find the combination that results in the best model performance.
- For each combination of hyperparameters in the grid, the model is trained and evaluated using cross-validation. The combination that yields the best performance metric (e.g., accuracy, F1 score) is selected as the optimal set of hyperparameters.

3.2.4 Grid Search for Hyperparameter Tuning:
- Grid search is employed to fine-tune hyperparameters of machine learning models, such as the number of neighbours in kNN or the maximum depth of a decision tree.
- By defining a grid of hyperparameter values, such as different values for k in kNN or tree depths in a decision tree, and using k-fold cross-validation, you can systematically search for the combination of hyperparameters that results in the highest accuracy. Grid search helps in optimising the model's performance without manually tuning hyperparameters.

Methodology

4.1 Data Pre-Processing

4.1.1 Train test split
The train-test split is a common practice in machine learning to evaluate the performance of a model. It involves splitting a dataset into two subsets: one for training the model and the other for testing its performance. The primary goal is to assess how well the model generalises to new, unseen data, as the purpose of machine learning models is to predict data, rather than learn the supplied data as accurately as possible.

- Training Set: The larger portion of the dataset is used for training the machine learning model.
  - This set is used to teach the model patterns and relationships in the data.
- Testing Set:
  - The smaller portion of the dataset is reserved for testing the model's performance. This set is used to evaluate how well the model can make predictions on new, unseen data.

The train-test split is crucial for:
- Performance Evaluation
  - It provides an unbiased evaluation of a model fit on the training dataset by testing its performance on a separate dataset.
- Overfitting Detection:

- It helps identify if a model is overfitting the training data by performing well on the training set but poorly on the test set. Overfitting means that the model is too accommodating to the bias in the dataset.

4.1.2 Feature scaling

Feature scaling is a crucial preprocessing step in many machine learning algorithms. It involves transforming the values of different features (variables) in the dataset to a standardised range. The main reasons for performing feature scaling include:

- Normalisation of Data:
  - Features may have different scales, and some machine learning algorithms are sensitive to the scale of the input features. Scaling ensures that all features have a similar scale, preventing certain features from dominating others.
- Distance-Based Algorithms:
  - Algorithms that rely on distances between data points, such as k-Nearest Neighbours (kNN), can be influenced by the scale of the features. Scaling ensures that the influence of each feature is more balanced.

- For this project, I have employed:
  - **Standardisation (Z-score normalisation):** Scales features to have a mean of 0 and a standard deviation of 1.
  - In Python, the StandardScaler from the sklearn.preprocessing module is commonly used for feature scaling. It standardises features by removing the mean and scaling to unit variance.

4.3 Cross-Validation and Hyper-parameter tuning

4.3.1 kNN Parameters

a. Distance Metrics

Used as a measure of closeness to find nearest data points

Commonly used:

- Euclidean Distance (or $L2$ Norm):

$$D(x,y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

- Represents the straight-line distance between two points in Euclidean space. It is sensitive to the magnitude of differences in individual dimensions.
- Suitable when the features have similar scales and when the spatial relationships between points are important.

- Manhattan Distance (or $L1$ Norm):

$$D(x,y) = \sum_{i=1}^{N} |x_i - y_i|$$

- Represents the sum of the absolute differences between corresponding coordinates. It is less sensitive to outliers than Euclidean distance.
- Suitable when the dataset has outliers or when differences along individual dimensions are more important than overall magnitude.

- Cosine Similarity:

$D(x,y) = 1 - \dfrac{\sum_{1=1}^{N} x_i \cdot y_i}{||x|| \cdot ||y||}$,where x and y are N-dimensional vectors.

- Measures the cosine of the angle between two vectors. It is commonly used for text data and is robust to differences in magnitudes.
- Suitable for high-dimensional data, sparse data (like text data), or situations where the magnitude of the vectors is not important.

i. *For the purposes of this investigation, I will be using **Euclidean** and **Manhattan** distance as they are most suitable in this context*

b. Neighbours

In k-Nearest Neighbours (kNN), the "k" value represents the number of nearest neighbours that are considered when making a prediction for a new data point (as mentioned in 3.1.1). It is a hyperparameter needs to be determined before training the kNN algorithm.

The choice of the "k" value can have a significant impact on the performance of the kNN model.

- Smaller k Values:
- Small value for k (e.g., k = 1), can result in a more flexible model that is sensitive to noise or outliers in the data. However, it may also lead to overfitting.
- Larger k Values:
- Larger value for k (e.g., k = 10), can result in a smoother decision boundary and a more robust model that is less sensitive to individual data points. However, it may lead to underfitting, especially if the dataset has complex patterns.
- For this project, we will determine the optimal value of k for balance between bias and variance, using Hyperparameter Tuning through k-fold cross-validation and grid search
- The range of k-values (1-20) is chosen based on general rule of thumb, since this parameter will be tuned later on.

4.3.2 Decision Tree Classifier Parameters

a. Maximum Depth of tree

- The maximum depth of the decision tree is the longest path between the root node and a leaf node. It represents the maximum number of levels in the tree.
- Controlling the maximum depth helps prevent the tree from becoming too deep and overly complex. A shallow tree may underfit the data, while a very deep tree may overfit the training data and perform poorly on new, unseen data.
- This parameter will also be tuned through cross-validation.

b. Minimum samples required to split a node

- The min_samples_split parameter sets the minimum number of samples required to split an internal node into child nodes. If the number of samples at a node is less than this parameter, the node will not be split, and it becomes a leaf node.
- It controls the granularity of the splits in the decision tree. Higher values ensure that a node is split only if it contains a sufficient number of samples, preventing small, noisy splits.
- This parameter will also be tuned to balance the trade-off between tree complexity and generalisation.

4.3.3 Cross-Validation function

GridSearchCV (from scratch)

- Iterates over model
  - For kNN: Euclidean/Manhattan and neighbours
  - For Decision Tree Classifier: max depth, min samples split
- Uses accuracy score as evaluation metric as discussed above in 2.2.2

4.3.4 Implementation - to find best hyperparameters

Models are implemented with best hyperparameters, and varying cross-validation folds. The results are as follows:

a. kNN

5-fold Cross-validation results
Best Hyperparameters: {'neighbours': 4, 'distance': 'euclidean'}
Best Cross-Validation Accuracy Score: 0.975

4-fold Cross-validation results
Best Hyperparameters: {'neighbours': 4, 'distance': 'euclidean'}
Best Cross-Validation Accuracy Score: 0.9750000000000001

3-fold Cross-validation results
Best Hyperparameters: {'neighbours': 14, 'distance': 'euclidean'}
Best Cross-Validation Accuracy Score: 0.9583333333333334

b. Decision Tree

5-fold Cross-validation results
Best Hyperparameters: {'max_depth': 5, 'min_samples_split': 2}
Best Cross-Validation Accuracy Score: 0.9416666666666668

4-fold Cross-validation results
Best Hyperparameters: {'max_depth': 4, 'min_samples_split': 2}
Best Cross-Validation Accuracy Score: 0.9583333333333334

3-fold Cross-validation results
Best Hyperparameters: {'max_depth': 6, 'min_samples_split': 2}
Best Cross-Validation Accuracy Score: 0.9500000000000001

Results

Based on highlighted values above, models where retrained and the following results were achieved:

Comparison with baseline and Analysis

Baseline Accuracy:

- Baseline Accuracy = 0.33

Model Results:

- **kNN Test Accuracy: 0.9333 (or 93.33%)**
- Confusion Matrix:
```
array([[12,  0,  0],
       [ 0,  7,  0],
       [ 0,  2,  9]])
```

- **Decision Tree Test Accuracy: 0.9333 (or 93.33%)**
- Confusion Matrix:
```
array([[12,  0,  0],
       [ 0,  7,  0],
       [ 0,  2,  9]])
```

Results Summary:

- Both kNN and Decision Tree models significantly outperform the baseline accuracy.
- The kNN and Decision Tree models exhibit similar and high test accuracies, suggesting robust performance.
- The variance in cross-validation accuracy for kNN and Decision Tree indicates the stability of the model across different folds and k values, with a 4-fold cross validation reflecting best predictive ability considering bias and variance.
- Similarity in confusion matrix also indicates similar interpretability of target across the two models

Evaluation

- **Model Performance:**
  - Both models have performed very well on this data, in achieving their target of accuracy in classifying the flower species with an accuracy of above 90%
  - However, based on my aim of finding the best ML model among the two, these results are not very helpful as they have both presented exactly the same accuracy scores
  - It could be deduced that both models work as effectively in classification for this dataset
- **Potential Improvements:**
  - Additional algorithms could be explored, such as Random Forest, Support Vector Machines.
  - Feature Engineering could improve model performance by determining the features that contribute more significantly to the classification to achieve higher accuracy scores.
  - This is verified by a study I found online, whereby a higher accuracy score was achieved with kNN for the same dataset through feature engineering (refer to feature selection in references)
- **Consideration of Dataset Size:**
  - Given the small dataset size (150 instances), this model may be overly complex, fitting well only for this dataset, and not generalise well to new data. High accuracy scores can be expected for a dataset of this size.
- **Domain knowledge:**
  - Seeking feedback from domain experts or stakeholders can provide valuable guidance on improving model performance or addressing specific challenges in the dataset.

Conclusions

The dataset is overly simple and does not consist of sufficient variance to effectively determine what ML model would fit the data better, as all the techniques employed in this project are carried out in a very minute scale. Furthermore, the nature of the dataset, being exactly balanced and well separated to begin with, also limit the potential for further analysis since the best metric for comparison is possibly only accuracy. However, on the basis of this data, both classification models kNN and Decision Trees are equally effective in achieving their goals.

References

- Feature selection: https://medium.com/nerd-for-tech/data-engineering-a-feature-selection-example-with-the-iris-dataset-11f0554e4b00