A Report

Submitted to

UNIVERSITY OF LIVERPOOL

Applied Artificial Intelligence
(COMP534)

Assignment-1
(Binary Classification)

BY

SUBHADEEP ROY
(201712383)

Introduction

Brief Overview of The Problem:

We have a dataset named "dataset_assignment1.csv" containing 700 rows and 10 columns. The columns are labelled as feature1, feature2, ..., feature9, and class. The class column has distinct values of 0 and 1, indicating the association of each entry or datapoint with one of these two classes. Our task is to train and deploy three machine learning classification models: KNN Classifier, Decision Tree Classifier, and Logistic Regression Model.

Python Libraries Used:

We employed several Python libraries in our analysis, utilising specific classes from each as outlined below:

Pandas: Utilized for reading the dataset stored in the .CSV format via the pd.read_csv() function, facilitating the conversion of data into DataFrame objects.

Matplotlib: Employed for generating feature versus count plots, enabling visualisation of the distribution of each feature within the dataset.

Seaborn: Utilised for generating histograms and kernel density estimation (KDE) plots for each feature present in the dataset, offering insights into their underlying distributions.

Scikit-learn (or sklearn): Leveraged for implementing various machine learning models, as well as for accessing evaluation metric functions, facilitating comprehensive analysis of model performance and comparison among different models.

Classification Methods and Hyperparameter Tuning: The classification techniques employed include Decision Tree classifier, K-Nearest Neighbour classifier, and Logistics Regression.

1. Decision Tree: To optimise the Decision Tree classifier, I utilised GridSearchCV to fine-tune its hyperparameters. The parameters subjected to tuning included- {'criterion': ['gini', 'entropy', 'log_loss'],'max_depth': range(1, 16), 'max_features': ['auto', 'sqrt', 'log2'], 'splitter': ['best', 'random']}. Here criterion is used to check the purity of the split, max_depth is used to prevent the model from overfitting, max_feature is used to know the number of features to consider when looking for the best split, spillter is used to choose the best split as each node.

The best parameters that I got after tuning the model – {'criterion': 'entropy', 'max_depth': 5, max_features': sqrt, 'splitter': 'random'}. Here we are using 'entropy' as criterion, as the dataset is also small. The max_depth is 5, which tells us that the model pruned the decision tree after branch 5. The max_feature is 'sqrt' (sqrt(n_features)), which tell us the number of features the tree uses to generate the nodes. The splitter is 'random' to choose the random best split. We can see in the decision tree, that our split starts using the feature 2, where the information gain is maximum. However, these parameters and the split change every time we perform the hyperparameter tuning.

2. K-Nearest Neighbour: The parameters I used to tune the model- { "n_neighbors":range(1,16), "algorithm":["auto", "ball_tree", "kd_tree", "brute"]}. Here n_neighbors is the k values of number to neighbours and algorithm is the method to compute the nearest neighbours.

The best parameters that I got using GridSearchCV are, n_neighbors or best k-value is 9 and algorithm is auto, as the dataset is small so time complexity shouldn't be a problem, otherwise we can use ball tree, kd tree, where time complexity is lesser.

3. Logistics Regression: The parameters I used to tune the model- {"penalty": ["l1","l2","elasticnet"], "C": [1,10,20,30], "solver": ["lbfgs", "liblinear", "newton-cg", "newton-cholesky", "sag", "saga"]}. Here penalty is to specify the norm of the penalty, where L1 is lasso regression, which is used for feature selection, L2 is ridge regression, which is used to prevent from overfitting and elastic net is the amalgamation of both L1 and L2. C is the inverse of regularisation strength, λ. Solver is the optimisation method, if we plot the cost function for logistic regression, we will get local minimums and global minimums, to overcome this problem we use different algorithms, which are listed above.

The best parameters I got using GridSearchCV are, l1 for penalty, C is 1, solver is liblinear, which makes sense as we don't have a large dataset.

Cross-validation using the KFold method was conducted, and hyperparameter tuning was performed using the GridSearchCV class. GridSearchCV was preferred over RandomSearchCV due to the relatively small size of the dataset, allowing for a comprehensive analysis of all hyperparameter combinations. This choice minimized the complexity associated with GridSearchCV.

Training Process:

1. Dataset was first divided into two parts – **X** being the independent/input features set from feature1 to feauture9 and **y** as the classification or target set with the "class" column.

2. X and y were further divided into the training dataset (80%) and the testing dataset (20%) using train_test_split.

3. GridSearchCv ad Cross validation were performed on the training dataset to find out the expected evaluation metrics score Now, for each classification algorithm, I used cross validation to train the model by using "cross_val_score" from sklearn, where cross validation is 5 and get the evaluation metrics including accuracy score, precision, recall in a pandas data-frame (**Table-1**).

Testing process: I used the models with best parameters which I got from the hyperparameter tuning stage and predict on the independent features (X_test) of test dataset and got the predicted values of dependent features (y_pred) for each classification algorithm, which I have used to get the evaluation metrics to report the performance of the algorithms, that I have discussed in the next section.

Table-1:

Cross Validation and Evaluation Metrics			
Metrics	**Decision Tree**	**KNN Classifier**	**Logistics Regression**
Accuracy	*0.9304*	*0.9696*	*0.9607*
Precision	*0.9220*	*0.9416*	*0.9454*
Recall	*0.9071*	*0.9471*	*0.9541*
F1 Score	*0.9227*	*0.9609*	*0.9587*

Evaluation Section

We can observe that the output feature – **'class'** has two categories **0** & **1.** We can consider category **0** as **True** and **1** as **False**, as the count of **0** is higher than **1**, this will help us to understand the confusion matrix in a better way for each of classification methods.

Definitions:

True Positive (TP): We predicted as positive (0) and it is indeed positive (0).

True Negative (TN): We predicted as negative (1) and it is indeed negative (1).

False Positive (FP): We predicted as positive (0) and it is turned out to be negative (1).

False Negative (FN): We predicted as negative (1) and it is turned out to be positive (0).

Evaluation Measures (Table2):

a. Accuracy: The proportion of correctly classified instances out of the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

b. Precision: It measures the proportion of true positive predictions out of all positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP}$$

c. Recall: It measures the proportion of true positive predictions out of all actual instances of a class in the dataset.

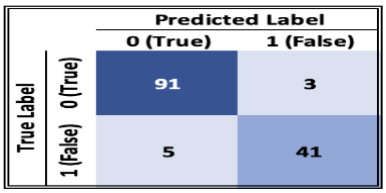
$$Recall = \frac{TP}{TP + FN}$$

d. F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics. It's particularly useful when the classes are imbalanced.

$$F\_1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

1. Decision Tree:

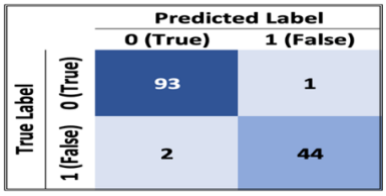
We can see the confusion matrix for decision tree classification method. Here true positive (**TP**) is 93, true negative (**TN**) is 41, false positive (**FP**) is 5 and false negative (**FN**) is 1.



Confusion Matrix for Decision

2. K-Nearest Neighbour:

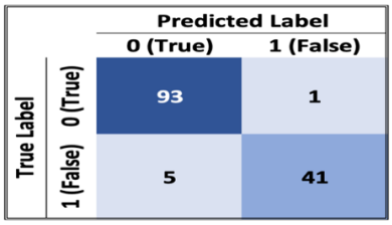
We can see the confusion matrix for KNN classification method. Here true positive (**TP**) is 93, true negative (**TN**) is 44, false positive (**FP**) is 2 and false negative (**FN**) is 1.



Confusion Matrix for KNN

3. Logistic Regression:

We can see the confusion matrix for Logistics Regression classification method. Here true positive (**TP**) is 93, true negative (**TN**) is 42, false positive (**FP**) is 5 and false negative (**FN**) is 1.

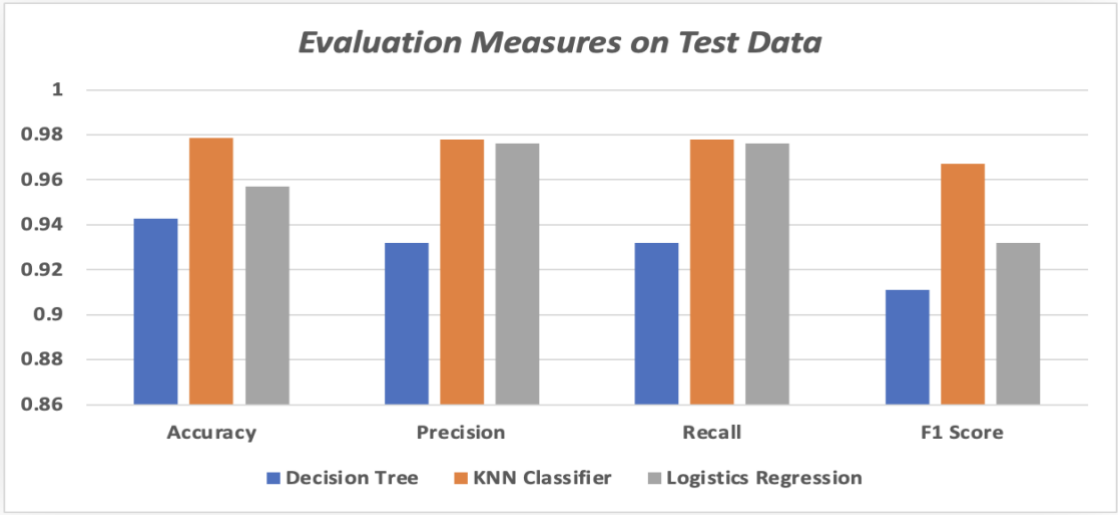


Confusion Matrix for Logistics Regression

Now, depending on the problem set, if **FP** is very important, we use precision score and **FN** is very important, we use recall score.

Table-2:

Evaluation Metrics for all of the Classification Methods			
Metrics	**Decision Tree**	**KNN Classifier**	**Logistics Regression**
Accuracy	*0.9429*	*0.9786*	*0.9571*
Precision	*0.9318*	*0.9778*	*0.9762*
Recall	*0.9318*	*0.9778*	*0.9762*
F1 Score	*0.9111*	*0.9670*	*0.9318*



Conclusion

Decision Made: K-Nearest Neighbour (KNN) achieved the highest accuracy at 97.86%, surpassing both Logistics Regression (95.71%) and Decision Tree (94.29%) . While Precision, Recall, and F1 Score exhibited similar performance across all three methods, KNN showed a slight advantage.

Decision Tree stands out for its transparency and ease of interpretation, accommodating various data types with minimal preprocessing. It effectively captures nonlinear relationships and is resilient to outliers, yet it's prone to overfitting and sensitive to minor data variations. Furthermore, it may demonstrate bias towards features with extensive levels or categories.

KNN, known for its simplicity and robustness, operates without assumptions regarding data distribution, making it suitable for diverse datasets. It excels in multiclass classification and naturally adapts to local data patterns, particularly beneficial for intricate decision boundaries. However, its computational efficiency becomes a concern with large datasets, and optimal performance relies on careful selection of distance metrics and neighbors (K). Additionally, KNN may face challenges with imbalanced datasets.

Logistic Regression offers simplicity, interpretability, and computational efficiency. It provides probabilistic interpretations of classification decisions and performs well with linearly separable data. It accommodates binary and multiclass classification problems and offers regularization techniques. Nonetheless, it assumes a linear relationship between independent variables and the log-odds of the target variable, which may not always hold true. Moreover, logistic regression struggles to capture complex nonlinear relationships and is sensitive to outliers and multicollinearity.

Overall, KNN demonstrates balanced performance across all metrics, making it the preferred choice for this classification task. Its superior performance compared to Decision Tree and Logistics Regression underscores its effectiveness for this specific dataset, solidifying KNN as the most effective classification method for this particular task.

Future Consideration: For enhanced evaluation, opting for a larger dataset would provide the models with more training data and deeper insights into the problem. Calculating feature correlations aids in identifying redundant data, optimizing model efficiency. KNN stands out due to its versatility and adaptability to multiclass problems, but consideration should be given to dataset size, balance, and quality, along with determining appropriate k-values or ranges. Additionally, employing principal component analysis (PCA) for dimensionality reduction can enhance model performance and data understanding. Removing outliers from features with outliers further refines model performance.