Machine Learning Assignment Report

PARNA PRATIM MITRA(G24AIT2015) & MD AZAM(G24AIT2073)

Q1: Data Preprocessing – Summary

1. In this portion, we had to perform data preprocessing on the `Cars93.csv` dataset, which contained 26 columns.

2. For the first question of assigning a type to each feature, we determined whether the features could be categorized in any order or not and then assigned the appropriate types.

3. Next, we handled missing values for three features. This was mostly done by using the median for numerical columns, and for non-numerical columns, we replaced missing values with "None."

4. Furthermore, noise was reduced by removing outliers and dropping irrelevant columns. We also used correlation to drop one column, as it carried almost the same information as another, making it redundant in the model. Keeping both could lead to multicollinearity, which makes it difficult to determine the independent effect of each variable on the target variable.

5. After that, to encode categorical values, we used different types of encoders: the Label Encoder for nominal features and the Ordinal Encoder for ordinal features.

6. Lastly, we normalized all the features using `StandardScaler` since it has less effect on outliers. The dataset was then split into training, validation, and test sets in the ratio of 70:20:10 as requested in the question.

Q2a: Linear Regression Task. - Summary

1. In this question, we worked with a dataset named `linear_regression_dataset.csv`, which contained two columns: Height and Weight.

2. First, we used the inbuilt Linear Regression model to predict the weight from the height by splitting the data into a 60:40 ratio for the training set and test set.

3. We inspected the coefficient obtained and then plotted a scatter plot to visualize the line of best fit.

4. As per the instructions, we built a Linear Regression model from scratch using the linear hypothesis function:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

   and used the gradient descent algorithm to find the optimal values of the parameters to minimize the prediction error.

5. We performed 2000 iterations with a learning rate of 0.05 and obtained results similar to the built-in model.

6. Finally, we visualized the results graphically using a scatter plot.

Q2b: Logistic Regression Task. - Summary

1. In this Logistic Regression exercise, we worked on a dataset named `logistic_regression_dataset.csv`, which contained 5 columns, each having 400 entries.

2. We visualized the data for any missing (null) entries but found none. Additionally, we encoded categorical columns like gender using One-HotEncoder. The `Id` column was dropped since it was not useful for the model.

3. We then split the data into a 70:30 ratio and trained the model using scikit-learn's inbuilt Logistic Regression model. After training, we made predictions on the test set, printed the accuracy and confusion matrix, and visualized the results graphically.

Q3: SVM. - Summary

1. In this SVM exercise, the dataset contained 14 columns, each containing 5000 non-null entries.

2. Out of 6 features, 3 features were selected, namely `Income`, `CCAvg`, and `Mortgage`, based on the correlation between them as well as with the target variable `Credit Card`. It was also observed that all the 6 features had a very insignificant correlation with the target variable, making the choice of features somewhat insignificant.

3. A 3D scatter plot was plotted to visualize the data. However, the plot showed a very low or insignificant relationship and randomness amongst the data points.

4. The dataset was split on an 80:20 basis. A LinearSVC model was applied to the test data with different values of the regularization parameter C, ranging from 0.0001 to 1000.

5. After making predictions, all relevant metrics such as the classification report, confusion matrix, and score were inspected.

6. GridSearchCV was then used to find the best value of C. Based on the optimal value of C, the model was retrained and predictions were made on the test data. The best possible score obtained was 0.586.

7. Although the final score is not particularly high, it is still reasonable given the choice of features, as the entire dataset exhibited almost no correlation among features except for the 3 chosen ones.

Q4: Decision Tree and Random Forest. - Summary

1. In the Decision Tree Exercise, the IRIS dataset was used. It contained 6 columns, each containing 150 non-null entries.

2. The `Id` column was dropped since all values were unique, making it irrelevant for classification purposes.

3. To visualize the dependency of `setosa`, `versicolor`, and `virginica` on sepal or petal length and width, KDE plots were made for all features. It was found that `PetalWidth` is a very good feature for splitting the data.

4. The target column, i.e., `Species`, contained non-numerical values, which were encoded using `LabelEncoder` as follows: 0 for `setosa`, 1 for `versicolor`, and 2 for `virginica`.

5. The entire cleaned dataset was fit using `DecisionTreeClassifier` from scikit-learn, and the tree was visualized. After making predictions, the decision tree produced very good classification results, which were evaluated using the confusion matrix and classification report.

6. The entire process of training and testing was also performed using `RandomForestClassifier` with `n_estimators` (number of trees) set to 500 and `max_depth` set to 2 to prevent overfitting. Better scores were obtained using the random forest model as with the decision tree.

7. Although random forests generally yield better scores than a single decision tree, since this was a small and simple dataset, the performance was almost same in both cases.