CITS5508 Machine Learning

Student Name: Varun Jain

Student Number: 21963986

Due: 26th March 2021

Question 1

Suppose that you are given a complex, multiclass classification problem and that your machine learning library only has the Support Vector classifier. Describe all the steps that you would take to train this classifier for your problem.

Step 1: Exploratory Data Analysis

1. Import the Pandas library and use the appropriate Pandas function to load the dataset into the system. For a csv file, use .read_csv() function.
2. Next, I would use the following Pandas function to gain insight about the structure and what sort of data we have to deal with.
   - .head(): returns the top 5 rows from the dataset
   - .columns(): returns a list of all the column names
   - .shape(): returns the total number of rows and columns
   - .info(): returns the datatype for each attribute and the number of non-values numbers.
   - .describe(): statistical summary of all the numerical attributes
   - .value_counts(): returns the total count for all unique values
3. Plot a histogram and scatter plot to visualise the data, and to see the correlation between the attributes.

Step 2: Data Preparation and Data Cleaning

1. For data inconsistency in the numerical attribute columns, we can compute the following methods:
   a. Remove the entire numerical attribute from the dataset if the dataset is missing a high percentage of its values.
   b. Replace the missing value with NaN (Not a Number) or some central value imputation such as mean, median, mode, using the imputer function.
   c. Remove the entire observation
2. For data inconsistency in the categorical attributes, we can compute the following:
   a. Use .value_count() to return the total number of instances for each unique class to ensure that all attributes has proper naming conventions, that there is no types duplicating the same class.
3. Define the independent (input feature) and dependent attributes (target variable)
4. For all dependent categorial attributes, use One Hot Encoding to create one binary attribute per class
5. Partition the dataset into 80:20 ratio set. 80% of the data is allocated to the training set and the remaining 20% is allocated to the test set.

Step 3: Data Transformation

1. Convert the numerical data types to one standard format, using the Pandas function .astype()
2. Apply feature scaling to standardise or normalise the data as Machine Learning Algorithms do not perform well on unscaled numerical data. In this instance, I would use the normalization method min_max() to resale the data.

Step 4: Train the Model

1. Perform multiclass classification using multiple binary classifiers. To train the binary classifier, we will use the one-verse-one approach with a Support Vector Machine (SVM) classifier.
2. For the SVM classifier, choose the appropriate hyperparameters to find the best estimator's value.
   a. Set random_state parameter to 42, this will ensure that the SVC classifiers picks the same random instances to ensure accurate results.
   b. Use the default kernel 'rbf', however for testing purposing we can compute the classifier against a 'linear', 'poly' and 'sigmoid' kernel using a GridSearchCV.

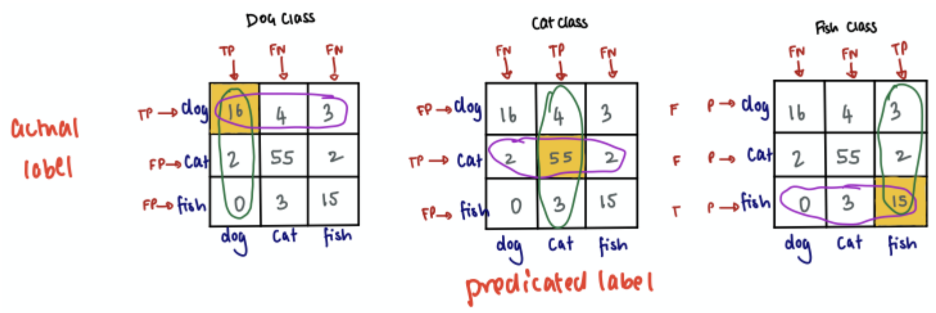
c. Predefine the regularisation term C from [0.01, 0.1, 1, 10, 100] with each value increasing with a magnitude of 10 and run this value with the rbf kernel in the GridSearchCV.
d. Include the Gamma hyperparameter. The Gamma values range from the default ‘scale value’ with an decrease and increase in the magnitude of 10.
e. Loop through the predefined hyperparameter values stated above using the GridSearchCV to find the best estimators to evaluate our model.

3. Fit your model using the .fit() method with the scaled training set data and the class.

Question 2

To calculate the average precision score and the average recall score, we have to first compute the precision and recall values for each class individually, then take the mean by computing the sum of score dividing it by the total number of classes.

Note Precision = $\frac{TP}{TP+FP}$ and Recall = $\frac{TP}{TP+FN}$. As we can diagram below, the False Positives and the False Negatives have been labelled on the confusion matrix for each class separately, on the vertical and horizontal axis retrospectively. The yellow cell indicates the true positive meaning all the classes that got identified correctly and with this, there are two distinct colour markers: purple and green. The green marker marks False Positive and True positive value for a certain class and the purple marker marks the True positive and False Negative values for a certain class.



In reference to the dog class confusion matrix, shown above, we can see that the classifier correctly identified 16 dogs. The recall score for dog is the number of dogs that got identified out of the actual dog class indicated by the purple line. The precision for the dog class is the number of correctly predicated dogs from the dog class indicated by the green line. In this case,

$$\text{Recall Score for dog class} = \frac{\textit{yellow cell}}{\textit{sum values inside the purple marker}}.$$

$$\text{Precision Score for the dog class} = \frac{\textit{yellow cell}}{\textit{sum values inside the green marker}}$$

The precision and recall can be calculated for the remaining two classes in a similar manner as the dog class.

Once, all the average precision and recall values have been calculated for each class

- Sum the precision value and divide it by the number of class, which in this case is three.
- Sum the recall values from every class and divide the total by three.

The Mathematically Computation as follows:

Precision for the class dog = $\frac{16}{16+2} = \frac{8}{9} = 0.89$

Precision for the class cat = $\frac{55}{62}$ =0.89

Precision for the class fish = $\frac{15}{20} = \frac{3}{4} = 0.75$

Average Precision for all three classes = $\frac{0.89 + 0.89+0.75}{3} = 0.8\ (1.dp)$

Recall for the class dog = $\frac{16}{16+4+3} = \frac{16}{23} = 0.70$

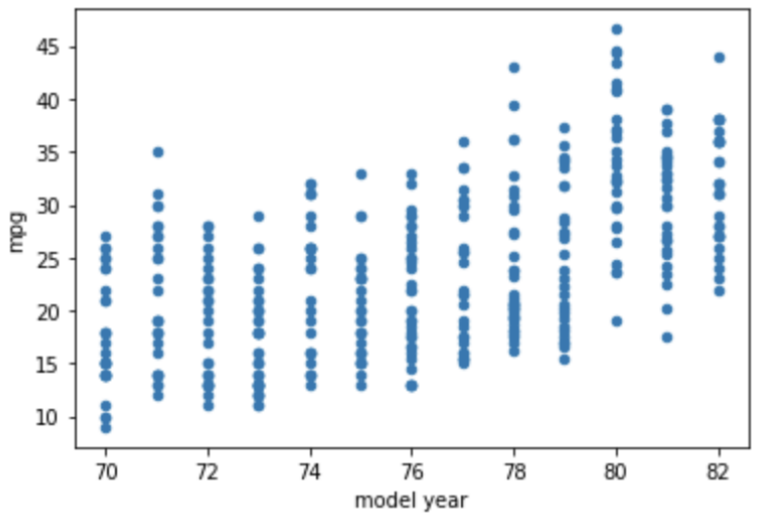
Recall for the class cat = $\frac{55}{59}$ =0.93

Recall for the class fish = $\frac{15}{18} = \frac{5}{6} = 0.83$

Average Recall for all three classes = $\frac{0.7+0.93+0.83}{3} = 0.8\ (1.dp)$

Question 3

1. Import the Pandas library as pd and use the read_csv() function to load the AutoMPG dataset.
2. Use the pandas function head() to inspect the dataset. We observe that the dataset is missing headers. So, we added appropriate header names to the respective columns in the data frame, which was provided to us by UCI ML Repository.
3. Use the pandas function .describe() to get a statistical summary of the data
4. Use the pandas function .info() to see the non-null value count and each attributes datatype. We notice that the attribute horsepower has been wrongly classified as an object, but it should be a float64 or int64.
5. Use the .unique() function on the horsepower attribute. We see that there is a missing value labelled "?".
6. Replace the missing value "?" with NaN (Not a Number) using the .replace() function.
7. Then convert the object datatype to a float64 using .astype() horsepower attribute
8. Plot a histogram to see the number of instances in each attribute
9. Remove the origin and car name attributes from the dataframe.
10. Compute the correlation to see how much each attribute correlates with the median. We observe that the model year has a fairly strong positive correlation with mpg. The new the car, higher the mpg.
11. Plot a scatter plot to see the correlation.
12. Plot a scatter plot mpg vs model year
13. Separate the independent and dependent variables. So, the y would store the mpg column and the rest of the data would be stored in x.
14. Split the model into two sets: a training set and a testing set.
15. Use the .shape() function to see the size of the split between the training and testing dataset.
16. Apply feature scaling to standardise the dataset. Standardization() bring the mean to zero and have a standard deviation of one.



From the scatter plot, I noticed that model year and mpg had a high correlation with each other. So for visualisation purposed, I created a scatter matrix for model year vs mpg to visualise the data in more depth.

Question 4

(i)

For large alpha values, the weightage on square of coefficient will be small as possible to minimise the cost function. Therefore, the slope of the line will flatten reducing the model's variance an increasing its bias. This model is subjected to under fitting due to the constrained placed on the coefficient magnitudes pushing the parameters close to zero, but not zero. Hence the model's complexity decreased.

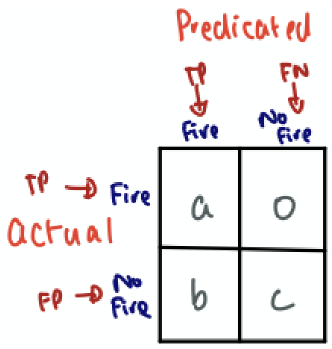
Lasso regression performs feature selection meaning it automatically selects the relevant features. For the lasso penalty, high alpha values push the coefficients all the way to zero. Therefore, it extracts features with the largest signal and eliminate the rest. This model is subject to underfitting. Low alpha values may result overfitting as there too many features.

(ii)

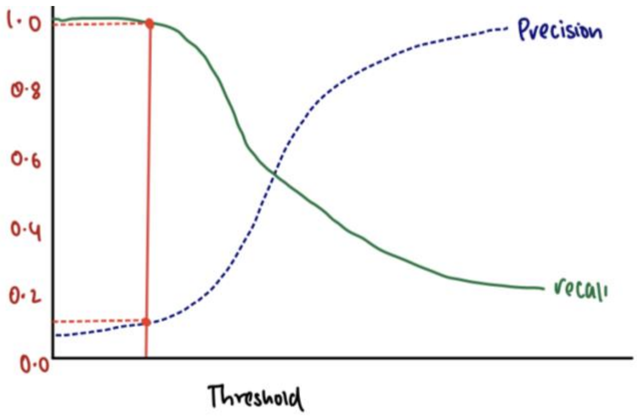
High Gamma values are prone to overfitting and amount of regularisation can prevent it. Gamma controls the influence of features meaning at higher gamma values, the features will be more in control of the decision boundary as the area of influence is smaller around individual instances. Thus, causing the decision boundary to be more irregular, and wiggling around individual instances. To counter the overfitting problem for both the RBF kernel and polynomial kernel, reduce the gamma value. For the RBF kernel, it will increase the width of the bell-shared curve meaning the decision boundary will smoothen out, and so allowing a larger range of area of influences for individual instances. For the polynomial, only the relevant features will be selected and eliminate the rest. Another way to reduce the overfitting for the polynomial is to use a lower degree polynomial meaning fewer parameters are selected by constraining the model and therefore, it smoothens the curve.

(iii)

1. Higher threshold values lower recall and increase precision, vice versa.
2. In the confusion matrix, you want zero false negatives. False negative basically means there you say there is no fire, but when in fact there is a fire. This can be shown on the confusion matrix below.



3. The decision threshold would be Maximum threshold before it dips below one, indicated by the red line on the diagram below.



Question 5

1.

a. An example of a **binary classification** is, given a car accident dataset, determine whether the accident was severe or not. Binary classifiers classify two classes.

b. An example of a **multiclass classification** is, the type of car: sedan, hatchback, sports wagon, and convertible. Multiclass classification has two or more classes.

c. Multilabel Classification is a classifier that is trained to classify objects such as in an image, the classifier will classify the objects that are present in the image such as car, tree, beach, sky, road.

d. An example of a **multioutput multiclass classification** is, the image of sports equipment. In this image, there are several different type balls such as cricket ball, basketball, soccer ball, and has other equipment's such as bats. In the class 'ball', we want to be able to classify the different type of balls correctly, whether it may be a soccer ball, cricket ball or a basketball.

2.

The first three lines of the code import the relevant library that are required to compute the following algorithm.

1. From the Scikit-Learn Library we have imported the following: linear_model and the metrics module with the SGDRegressor Class and Mean Squared Error Class (MSE), retrospectively.
2. We import the Pplot from the Matplotylib visualisation package and save as plt.

The following defines the Scikit-Learns SGDRegressor class with hyperparameters. The values are stored in a variable called sgd. The following hyperparameters are:

- **max_iter = 1** – change the default number of iterations on the training data to one
- **top = -np.infinity** – specify the tolerance for the stopping criteria to negative infinity
- **warm_start = True** – any previously trained fitted models are used to initialise the new model on the second call.
- **penalty = 'None'** – no regularization is applied
- **learning_rate = 'constant'** – the step size of each iteration is constant
- **eta0=0.005** – the initial learning rate is set to 0.005

The following parameters are initialised: The following variables will store the lowest MSE value, the best iteration of the SCG with the lowest MSE value.

- **min_val_err = np.float("inf")** – stores a positive infinite float number.
- **best_epoch = None** – stores the iteration number of the for loop when the lowest MSE is found
- **best_sgd = None** – records the SGD value with the lowest MSE.

Looking at the for loop, previously under the SDGRegressor, the max number of iterations was set to one. However, SDGRegressor the number of iterations have been manually set, and now it will loop through the training data 500 times. For each iteration, the algorithm will fit the model on the training data. We will predict the validation data using the trained model and then compute the MSE for the predicated data, to measure the accuracy. MSE value will be stored in a variable named err. The if statement will compare the MSE value of the current iteration with the lowest MSE value recorded. If a lower MSE value is found, then min_val_err, best_epoch and best_sdg will be updated with the MSE value, the index number and sgd value, retrospectively. Otherwise, it will dsregard that iteration and move onto the next, After the loop has finished, the algorithm will print out the best epoch i.e. the index with the lowest MSE value. Then will predict the validation set with the best SDGregressor, best_sgd. The value computed will be stored in a variable call y_pred, which it then will be used to MSE value and print it out onto the screen.

Lastly, this is where the visualization package comes into play.

1. Plots the regression line between the independent and dependent variables
2. Plots the regression line between the independent and predicted variables
3. Two legends will be added to plot, ground truth and predication.