CSE3CI: Computational Intelligence

Assignment 1: Report

Student Name: James Worcester  
Student Number: 20767086

## **Load the dataset, do basic data preprocessing, and split the dataset.**

**In your report, describe the dataset (before and after the processing), describe how you did the below steps (you just need to describe the steps and the result you get from each step, do not attach codes in the report), and discuss why you did them.**

* 1. **Describe the dataset (before and after the pre-processing), for example, variable type, and data shape. (You may also consider applying correlation analysis, optional)**

Before pre-processing, I initially I loaded the ‘income.csv’ dataset into pandas before executing a few commands in order to properly describe the dataset. First, I checked the dataset length to see that it had 26,215 records, and then ran describe() to get a summary of the numerical columns in the dataset (income, age & hours-per-week). Next I checked the dataset shape to see that it had a shape of 10 columns and 26215 rows, before checking the data types for each variable, allowing me to ascertain that income, age and hours-per-week all had data types of ‘int64’, whilst each other variable had a type of ‘object’. I then performed a simple correlation analysis of the numerical columns to give a brief overview of correlations in the data.

After pre-processing, I checked the dataset length to determine it had shrunk to 21,537 records, before describing the database to be able to view various metrics about each variable. I also checked the dataset shape to verify that the number of rows and columns had changed, and then checked the data types for each of the 41 columns. Finally, I performed a simple correlation analysis to be able to view correlations in the data

* 1. **Deal with missing values (if there are any) and use a proper method to handle categorical variables.**

I dealt with missing values by first identifying that the workclass variable had 1396 missing values and that the occupation variable had 1401 missing values. To deal with the missing values, I dropped the rows that had those missing values, and checked the database length to see that it had reduced to 24,814 records. Categorical variables were handled and dealt with in part d.

* 1. **Remove duplicated inputs if there are any**

I determined that there were duplicated rows in the dataset, and was thus able to remove those duplicated rows, before checking that the database length had reduced to 21,537.

* 1. **Handle the categorical variables.**
     1. **For the ordinal variable education, assign values 1 to 16 to the categories in this order: Preschool, 1st-4th, 5th-6th, 7th-8th, 9th, 10th, 11th, 12th, HS-grad, Some-college, Assoc-voc, Assoc-acdm, Bachelors, Masters, Prof-school, Doctorate**

To handle the categorical variables, I checked the dataset data types again, before renaming ‘marital-status’ to ‘marital\_status’ to better suit pandas, then checking the value count for each of the seven categorical variables. According the specification above, I assigned values 1 to 16 to each of the education categories in place of their string values.

* + 1. **For the binary variable sex, assign value 0 to Male and value 1 to Female**

For the binary variable sex, I replaced the values of Male and Female with 0 and 1 respectively.

* + 1. **For the rest of the variables, apply dummy coding to deal with them**

For the remaining categorical variables of ‘workclass’, ‘marital\_status’, ‘occupation’, ‘relationship’ and ‘race’ I applied dummy variables for each permutation, before checking the head of the database and checking to see that the variable data types had changed and updated appropriately. I also checked the database shape to verify that additional database rows had been created for each dummy variable.

* 1. **Split the dataset into training and testing (with 10% of the dataset for testing).**

Before splitting the database, I assigned the target variable of income to y, and assigned all other input variables to X. I then split the training and testing dataset using sklearn and train\_test\_split, splitting the dataset into X\_train, X\_test, y\_train and y\_test, making sure that the test size was 10% of the dataset, and that a consistent random state of 123 was set.

* 1. **Apply normalisation on X (both training and test set).**

I applied normalisation to both the training and test sets using sklearn and MinMaxScaler().

1. **Train and evaluate the 2 classification models on the training set with the cross-validation method, optimize the models and evaluate models on the test set.**

**In your report, you need to start by explaining the basics of Logistic Regression and SVM. Then, describe the cross-validated and test results from the two models with default parameter settings, and compare and discuss the results among models. Next, describe what steps you have taken for finetuning your model (changing the parameters), describe the parameter settings that you applied in finetuning, and compare the results for each model (before and after finetuning for each model). Finally, compare the evaluation results across two optimised models on the test set, and discuss your findings. (You may consider using a table to record all the modelling results)**

Logistic Regression estimates the probability of a binary event occurring based on one or more independent variables (predictors). As Logistic Regression involves Binary Classification, the aim is to divide the dataset with an n-dimensional hyperplane between the target variable values of 0 and 1 (or positive 1 and negative 1) whilst minimising any classification errors by using different Gradient Descent and Loss Function methods to attempt to accurately predict whether the target value is a 0 or 1 (positive or negative) based on predictors.

Whereas Logistic Regression will try to separate the dataset by an arbitrary hyperplane, a Support Vector Machine (SVM) aims to create the ideal n-dimensional hyperplane that maximises the margin (separation) between the positive and negative samples so that the probability of a binary event occurring can be accurately estimated. If the data cannot be completely separated because a constraint has been violated (classification error), the violating data points will be penalised by the SVM.

Both Logistic Regression models and SVMs train and test data to find the most accurate machine learning model for a given dataset.

* 1. **Define the two regression models, including Logistic Regression, and SVM, with their default settings**

I defined the two regression models by importing ‘LogisticRegression’ and ‘SVC’ from sklearn and used their default settings to print out a testing accuracy of 0.8101207056638812

for Logistic Regression and 0.7957288765088208 for SVM. It’s important to note that because the default ‘lbfgs’ solver was used for Logistic Regression it was unable to converge within the default number of iterations and displayed an error message.

* 1. **Define 10-fold cross-validation to train and evaluate the two models based on the average score**

Firstly I defined a 10-fold cross-validation with data shuffling, ran that 10-fold cross-validation on both models using default parameter settings and determined that the average accuracy scores based on the cross-validation results were 0.8050359227863669 for Logistic Regression and 0.7969871056916021 for SVM.

Both models compare to the 2a testing accuracy scores and to each other very well, showing a low variance between models. It’s important to note that because the default ‘lbfgs’ solver was used for Logistic Regression it was unable to converge within the default number of iterations and displayed an error message.

* 1. **Apply parameter finetuning steps to the two models separately to optimise the model performances and compare the cross-validated results before and after finetuning for each model.**

To finetune my logistic regression model I initially used the following parameter settings:

**Penalty: [l1, l2]**

**C: [1, 10]**

**random\_state: [123]**

**solver: [saga, liblinear]**

After checking the parameter setting for the best selected model I finetuned my logistic regression model parameters to:

**Penalty: [l2]**

**C: [1]**

**solver: [liblinear]**

Before finetuning, the Logistic Regression model achieved an accuracy of 0.8050359227863669 and after achieved an accuracy of 0.8053456001439147.

To finetune my SVM model I initially used the following parameter settings:

**kernel: [linear, poly]**

**C: [1, 10]**

**degree: [3, 8]**

**gamma: [auto, scale]**

After checking the parameter setting for the best selected model I finetuned my SVM model parameters to:

**kernel: [poly]**

**C: [10]**

**degree: [3]**

**gamma: [scale]**

Before finetuning, the SVM model achieved and accuracy of 0.7969871056916021 and after achieved an accuracy of 0.8045709676612425

* 1. **Evaluate the two optimised models (with the best parameter setting from the above step for each model type) on the test set, and compare the results with what you got from 2b**

When testing on the test set, the Linear Regression model achieved an accuracy of 0.8105849582172702 compared to an average accuracy of 0.8050359227863669 from the 10-fold cross-validation in question 2b. Similarly, when testing on the test set the SVM model achieved an accuracy of 0.8045496750232126 compared to an average accuracy of 0.7969871056916021. These findings show that both models performed more accurately once optimised.

1. **Apply K Means clustering on the normalised training input X, and understand the grouping of training data by investigating the prototype from each cluster**

**In your report you need to start by explaining the basics of the K means method. Then describe how many clusters you have chosen in your data clustering and how many data samples have been assigned to each cluster with the K means model. Compare the differences and similarities between the prototype for each cluster. And finally, evaluate the accuracy of the clustering method based on the testing set, and compare the results from the 2 models in 2d, and discuss your findings.**

The K means method involves unsupervised learning that requires data but no labels to detect patterns. However, whilst useful when you don’t know what you’re looking for, it can return garbled results. K means tries to put all data points into multiple clusters (in this case two clusters for income <= $50k and income > $50k) in order to predict class labels, and does so by first choosing the number of clusters, then selecting random points from the data as centroids before assigning all the points to their closest cluster centroid, recomputing the centroids of the newly formed clusters and repeating the assigning of points to the closest cluster centroid and recomputing the centroids of each cluster until a stopping criteria is reached.

* 1. **Apply clustering on the normalised training input X (you can determine the number of clusters by considering how many classes for the target y)**

I first normalised the training input X before applying kmeans clustering on it by defining the random state number as 0 and the number of clusters as two since the target variable income has two possible values of 0 or 1.

* 1. **Identify how many data samples have been assigned to each cluster.**

I identified that 10,265 data samples had been assigned to cluster 0, and 11,272 had been assigned to cluster 1, for a total of 21,537. For a sanity check I compared the total value to the shape of the dataset that also contained 21,537 records.

* 1. **Extract a prototype from each cluster and investigate their similarity and difference**

After extracting a prototype from each cluster, we can investigate and compare the differences and similarities between each prototype:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| income |  |  |
| age |  |  |
| education |  |  |
| Sex |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 1. **Evaluate the clustering accuracy with the testing set and compare with the results from 2d.**