ECE9603/9063: Data Analytics Foundations

by

Omar Awadallah

oawadall@uwo.ca

Instructor:

Dr. Katarina Grolinger

Department of Electrical and Computer Engineering

Western University

London, Ontario, Canada

Table of Contents

1. General Overview

1.1 Problem Description

The importance of cars and automobiles evolves continuously as the economy grows and job opportunities increase. Cars provide more freedom for mobility and significantly decrease travel times. Therefore, car sales are important as demand increases and these sales are often an indicator of the economic growth in countries.

The problem that is discussed in this report aims to predict whether a specific person is most likely to buy a car or not based on their information. Predicting the likelihood of customers to purchase cars helps companies to identify which groups of people are least likely to buy cars and target their offers and advertisements towards this group of people, it also helps companies to develop offers that suit a specific group based on their age, gender, or salary. A dataset that contains information about the previous customers for a company and whether they purchased a car or not is used to solve this problem.

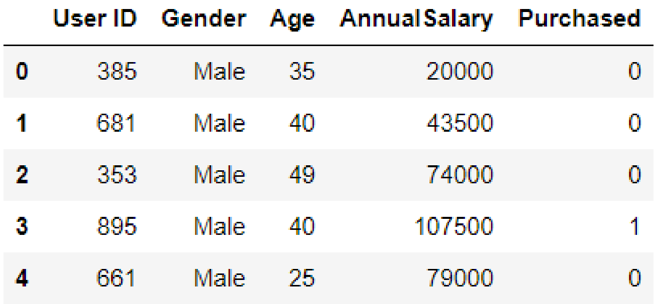
1.2 Data Description

The dataset that is used in this problem is obtained from Kaggle [1]. The dataset contains five different attributes; the customer or user ID that ranges from 1 to 1000, the gender of the customer (male/female), the age of the customer that ranges from 18 to 63, the annual salary for the customer that ranges from 15,000 to 153,000, and a *Purchased* attribute that specifies whether the customer has bought a car or not, the values 0 and 1 represent the *purchased* and *not-purchased* classes for this attribute. The dataset consists of five columns and 1000 rows where each row represents a previous customer and their personal information. In this problem, all 1000 samples and the *Gender, Age, Annual Salary,* and *Purchased* attributes are used, the *User ID* is not used. Table 1 shows a summary of the dataset attributes, and Figure 1 shows the first five rows of the dataset.

Table 1: Overview of initial data attributes

	Description	**Type**	**Values**
User ID	The ID for a customer.	Integer	1 to 1000
Gender	The gender of the customer	String	Male/Female

Age	The age of the customer.	Integer	18 to 63
Annual Salary	The customer's total salary in a year.	Integer	15,000 to 153,000
Purchased	The value represents whether a specific customer has bought a car or not.	Integer	0 or 1



	User ID	Gender	Age	AnnualSalary	Purchased
0	385	Male	35	20000	0
1	681	Male	40	43500	0
2	353	Male	49	74000	0
3	895	Male	40	107500	1
4	661	Male	25	79000	0

Figure 1. Initial dataset

2. Background of Neural Network

Three different architectures were considered to solve the problem outlined in section 1.1, the Multi-Layer Perceptron (MLP), The Recurrent Neural Network (RNN), and the Long Short-Term Memory (LSTM). MLP architecture learns to map inputs to outputs, this flexibility allows MLP to map different inputs after learning to one output that represents a specific class which makes it good for classification problems. RNN architecture is sequential which means it uses sequential data to solve sequential prediction problems. Time-series data or sequences of characters are types of sequential data that RNN uses to forecast future values or perform Natural Language Processing (NLP). LSTM architecture is designed to process regression and classification problems based on time-series data. Therefore, MLP architecture is used to solve the problem in this paper because it better processes non-sequential and non-time-series data.

MLP architecture consists of an input layer, multiple hidden layers, and an output layer. Each layer consists of several neurons, each neuron in the input layer is connected to each neuron in the next hidden layer, and each neuron in the hidden layer is connected to each neuron in the next hidden layer or the output layer. Neurons are nodes that receive one or more input signals from the input data or the previous layer, these signals are transmitted only in the forward direction, and there are no loops which means signals cannot be transmitted back to the previous layers, MLP is also known as Feed Forward Neural Network (FFNN) due to this behavior of forward signal transmission [2]. Figure 2 shows a three-layer MLP.

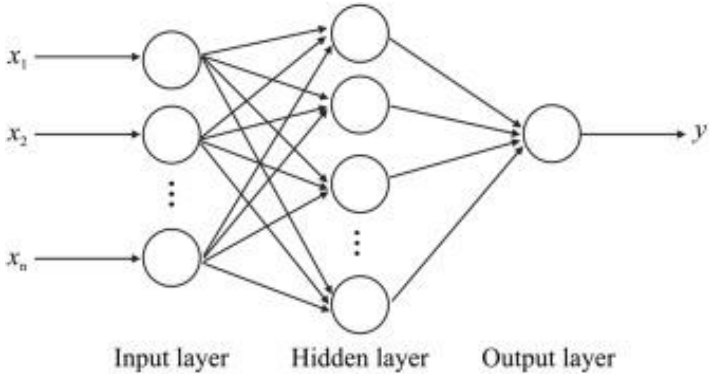


Figure 2. Three-layer MLP architecture

For each neuron in the hidden layers, an activation function needs to be calculated. Each neuron has a specific weight (*w_{ij})*, where *i* is the *ith* neuron in the hidden layer and *j* is the *jth* hidden layer in the network. Each weight is multiplied by the input from the input layer or the input from the previous hidden layer if there is more than one hidden layer, this is represented in Equation 1. In most cases, the L2 regularization term is applied which shrinks the weights to smaller values to prevent overfitting.

$$z = \sum_i w_i x_i + b \tag{1}$$

Where:

- w_i, weight
- x_i, inputs
- b, L2 regularization term

After the weights are multiplied with the inputs, the results are fed into an activation function which decides whether a neuron is important or not and should be activated or not. The activation function is a mathematical operation, in the MLP used in this problem, the three possible activation functions are *Logistic, Tanh,* and *ReLU* functions. Figure 3 shows the three different activation functions available in MLP from *scikit-learn.*

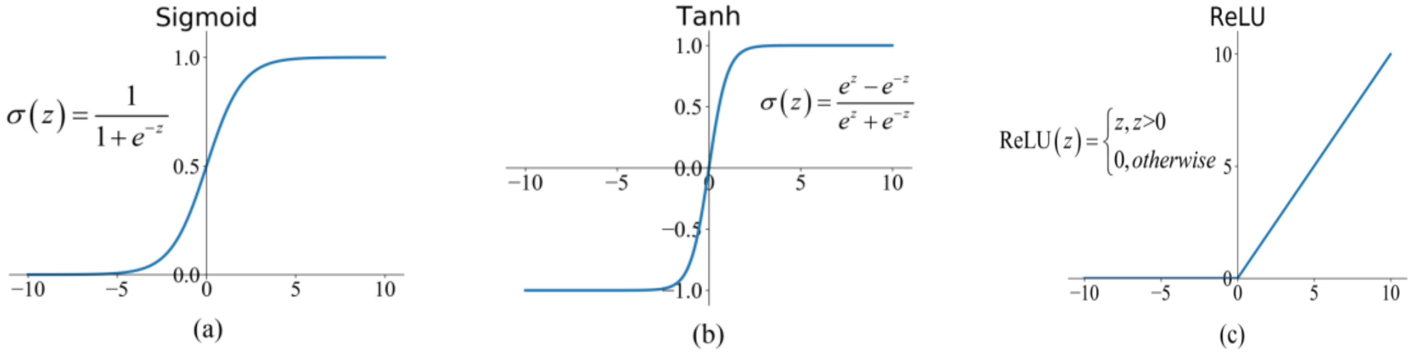


Figure 3. Logistic, Tanh, and ReLU activation functions [3]

The MLP also has a hyperparameter called *solver*, this parameter specifies the type of solver for weight optimization, and it has three different solvers, *lbfg, sgd,* and *adam.* The *sgd* solver is chosen for this parameter and it stands for Stochastic Gradient Descent which is a technique that picks a random part of training data at each training step and computes the gradient. The gradient descent is used to minimize the loss function by finding the local minimum, it uses a *learning rate* which is another hyperparameter in MLP. The learning rate specifies how large the steps that the gradient descent takes to find the local minimum, a higher learning rate will roll faster down the slope, while a lower learning rate rolls slower down the slope.

To conclude, the number of layers including hidden layers and the number of neurons is an important hyperparameter that affects the network operation, this hyperparameter is denoted as *hidden_layer_sizes.* The L2 regularization term is another important parameter and is denoted as *alpha.* The type of activation function affects the decision to activate the neurons or not which also affects the network, this hyperparameter is denoted as *activation.* The type of solver is responsible for weight optimization and the learning rate is responsible for how fast the weights are updated which affects the network operation, these two parameters are denoted as *solver* and *learning_rate_init* respectively. Table 2 summarizes the important hyperparameters that affect the MLP network and which parameters are tuned.

Table 2: Summary of Hyperparameters

	Description	Type	Used	Tuned	Value
Hidden_layer_sizes	An ith element representing the number of neurons in the ith hidden layer	Tuple	Yes	Yes	Five hidden layers. Neurons in each hidden layer either 25 or 50.
activation	Activation function for hidden layers	String	Yes	Yes	Logistic, tanh, or ReLU
solver	Solver for weight optimization	String	Yes	No	SGD
alpha	Strength of L2 regularization term	Float	Yes	Yes	(0.0001, 0.001, 0.01, 0.1)
Learning_rate_init	Step size to update weights	Float	Yes	Yes	(0.0001, 0.001, 0.01, 0.1)

The number of hidden layers used is five, one input layer and one output layer, for a total of seven layers. The number of neurons in each hidden layer is tuned to either 25 or 50, the combination for the number of neurons is as follows:

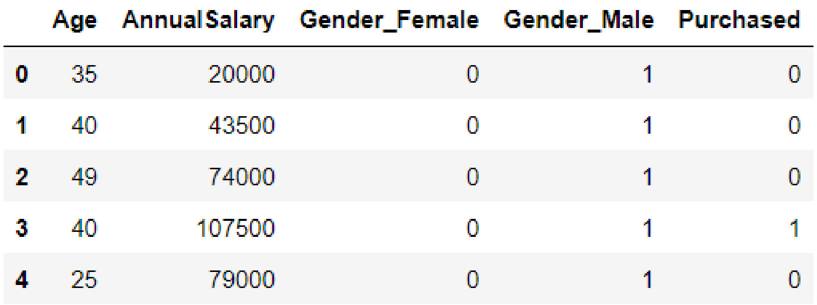
(25,25,25,25,25), (25,25,25,25,50), (25,25,25,50,25), (25,25,25,50,50), ……, (50,50,50,50,50)

The activation function is tuned to either *Logistic, Tanh*, or *ReLU*. The solver is set to the Stochastic Gradient Descent (*sgd*). Both the *alpha* and *learning_rate_init* parameters are tuned to either *0.0001, 0.001, 0.01,* or *0.1*.

3. Methodology

3.1 Data Pre-processing

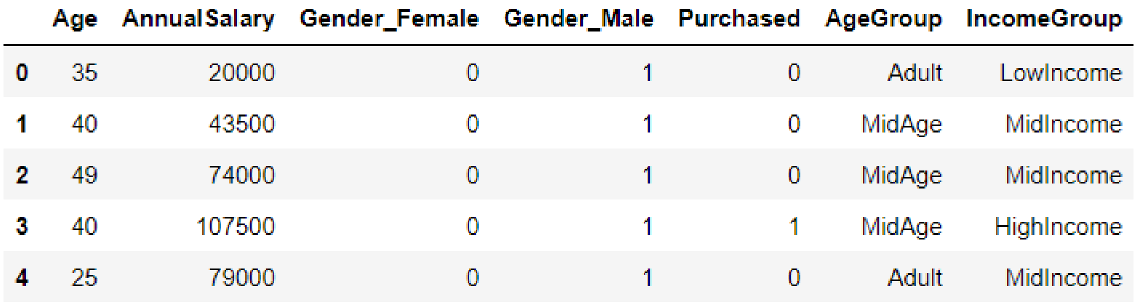
As mentioned in section 1.2, the information about customers are the user ID, gender, age, annual salary, and purchase condition. The *User ID* attribute is dropped from the data since it does not contribute to whether a person purchases a car or not, it is only an identifier for the customer in the company. The *gender* attribute is in string format and represents either male or female, hence one-hot encoding for this attribute is performed to create two numeric features representing whether the customer is male or female. Figure 4 shows the dataset after the first pre-processing step.



	Age	AnnualSalary	Gender_Female	Gender_Male	Purchased
0	35	20000	0	1	0
1	40	43500	0	1	0
2	49	74000	0	1	0
3	40	107500	0	1	1
4	25	79000	0	1	0

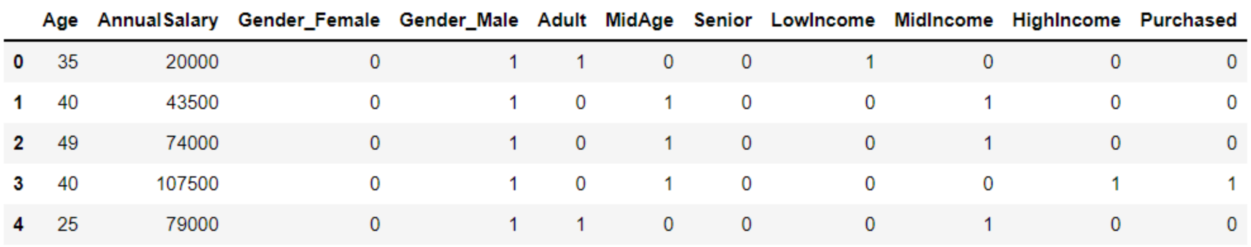
Figure 4. Dataset after the first pre-processing step

The next pre-processing step is to generate features from the *Age* and *Annual Salary* attributes. As mentioned in Table 1, the age ranges from 18 to 63 and this attribute is split into three groups, *Adult* (18-39), *Mid-Age* (40-59), and *Senior* (60+), a new feature called *AgeGroup* is created for these classes. Similarly, the salary ranges from 15,000 to 153,000 and this attribute is split into three groups, *Low income* (< 32,000), *Mid-Income* (32,000-99,999), and *High Income* (100,000+), a new feature called *IncomeGroup* is created for these classes. After generating *AgeGroup* and *IncomeGroup* features, one-hot encoding is used again to create numeric features that represent each group. The final dataset contains 1000 samples and 11 different attributes with 10 inputs and one output (*Purchased* attribute), Figures 5 and 6 show feature generation and one-hot encoding results respectively.



	Age	AnnualSalary	Gender_Female	Gender_Male	Purchased	AgeGroup	IncomeGroup
0	35	20000	0	1	0	Adult	LowIncome
1	40	43500	0	1	0	MidAge	MidIncome
2	49	74000	0	1	0	MidAge	MidIncome
3	40	107500	0	1	1	MidAge	HighIncome
4	25	79000	0	1	0	Adult	MidIncome

Figure 5. Dataset after feature generation for age and income groups



	Age	AnnualSalary	Gender_Female	Gender_Male	Adult	MidAge	Senior	LowIncome	MidIncome	HighIncome	Purchased
0	35	20000	0	1	1	0	0	1	0	0	0
1	40	43500	0	1	0	1	0	0	1	0	0
2	49	74000	0	1	0	1	0	0	1	0	0
3	40	107500	0	1	0	1	0	0	0	1	1
4	25	79000	0	1	1	0	0	0	1	0	0

Figure 6. Dataset after generating numeric features for age and income groups

3.2 Model Training

Before performing hyperparameters tuning and training the model, the full dataset with 1000 samples is split into training, validation, and testing sets, with 70% of the samples (700) for training, 20% of the samples (200) for validation, and 10% of the samples (100) for testing. The training set is used to train the model, the validation set is used to evaluate the model for each iteration when tuning hyperparameters to choose the best set of parameters, and the testing set is used to test the model after training is done using the best parameters. After splitting data, the inputs for training, validation, and testing sets are normalized using z-normalization, the outputs are not normalized because they are discrete categorical labels of 0 and 1. Therefore, denormalization is not needed to reverse the predictions. Equation 2 shows the z-normalization formula where *x* is the actual value of the input, µ is the training mean and *σ* is the training standard deviation.

$$z = \frac{x - \mu}{\sigma} \tag{2}$$

After inputs are normalized, the model is fitted on the normalized training inputs and the training outputs, the prediction is performed on the normalized testing inputs. However, before fitting and predicting, hyperparameters from Table 2 are tuned using Grid Search Cross-Validation (*GridSearchCV*). A parameter grid dictionary of the parameter's ranges that are specified in Table 2 is created, grid search with cross-validation tries all combinations in the parameter grid dictionary and evaluates every combination using k-fold cross-validation by splitting training set into *k* subsets, with *k-1* sets for training and *kth* set for validation. A 5-fold cross-validation grid search is used to evaluate all parameter combinations.

Once hyperparameter tuning is done, the tuned model with the best set of parameters is used for training on the dataset. The training step is performed over 150 epochs with a batch size of 128 samples for each epoch, the tuned model is fitted and the training and validation accuracy scores at each epoch are computed using the normalized inputs and the outputs of the training and validation sets, denoted as *Xtrain_norm*, *y_train*, *Xval_norm*, and *y_val*. A plot for the accuracy measures at each epoch achieved while training the tuned model is generated to observe whether the tuned model is learning or not. After training is done, the tuned model predicts outputs on the normalized inputs of the testing set, and the predictions are used for evaluation.

3.3 Evaluation Procedure

The tuned MLP model is compared with the Stochastic Gradient Descent Classifier model (SGD Classifier). The SGD Classifier model is used with its default parameters and hyperparameters tuning is not performed for this model. The models are validated before training using 5-fold cross-validation with *accuracy* as the scoring metric. The first step of the evaluation process is to compare the plots of the accuracy measures during the training process of both models to observe which model trains better on the data. The second evaluation step is to compute performance metrics for each model based on the 100 predictions performed on the 100 testing samples. Performance metrics include accuracy, precision, recall, sensitivity, and specificity. Accuracy is the number of correct predictions over the whole sample. Precision is the value that shows how many of the predictions labeled as positive belong to the positive class. Recall and sensitivity represent the number of predictions correctly labeled as positive out of all the positives in the data. Specificity represents the number of predictions correctly labeled as negative out of all the negatives in the data. All performance metrics are calculated using four variables; true positives (TP) which are predictions labeled as 1 and the true value is also 1, true negatives (TN) which are predictions labeled as 0 and the true value is also 0, false positives (FP) which are predictions labeled as 1 but the true value is 0, and false negatives (FN) which are predictions labeled as 0 but the true value is 1. The confusion matrices for the tuned MLP model and the SGD Classifier model are generated using the TP, TN, FP, and FN values. Equations 3, 4, 5, and 6 show the equations for accuracy, precision, recall, sensitivity, and specificity respectively.

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3}$$

$$precision = \frac{TP}{TP + FP} \tag{4}$$

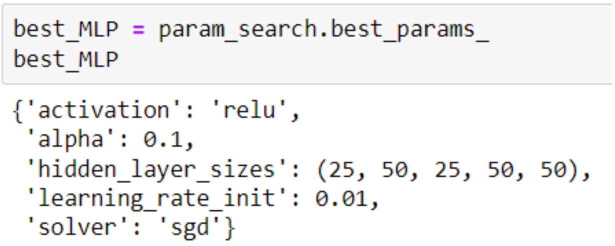
$$recall = sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$specificity = \frac{TN}{TN + FP} \tag{6}$$

The third and final evaluation step is to generate the receiver operating characteristic (ROC) and Precision-Recall (PRC) curves for both models. The ROC curve shows the trade-off between the specificity and sensitivity values. When the ROC curve is closer to the top left corner of the plot, the performance of the model is better. The PRC curve shows the relationship between the precision and recall values. When the area under the curve (AUC) in the PRC curve is bigger, the performance of the model is better.

4. Results and Discussions

The hyperparameter tuning using grid search found the best set of parameters to use to train the tuned model to be as follows:

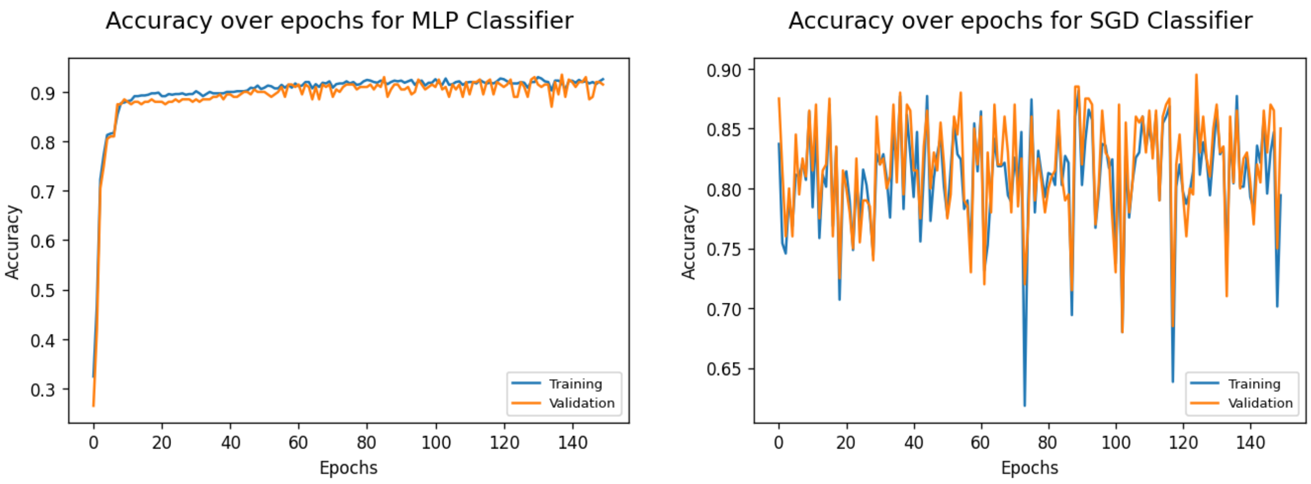


```
best_MLP = param_search.best_params_
best_MLP
```

```
{'activation': 'relu',
 'alpha': 0.1,
 'hidden_layer_sizes': (25, 50, 25, 50, 50),
 'learning_rate_init': 0.01,
 'solver': 'sgd'}
```

Figure 7. Best set of parameters from the hyperparameter tuning process

The best activation function is the *ReLU* function, with five hidden layers, 25 neurons in the first and third layer, 50 neurons in the second, fourth, and fifth hidden layers, an alpha value of 0.1, a learning rate value of 0.01, and an *SGD* function for the solver. The best set of parameters is used to train the tuned model on the normalized inputs of the training set and predict using the normalized inputs of the testing set. The training and validation accuracy measures during the training process of the tuned model over 150 epochs, for the MLP and SGD models, are seen in Figure 8.



(a) (b)

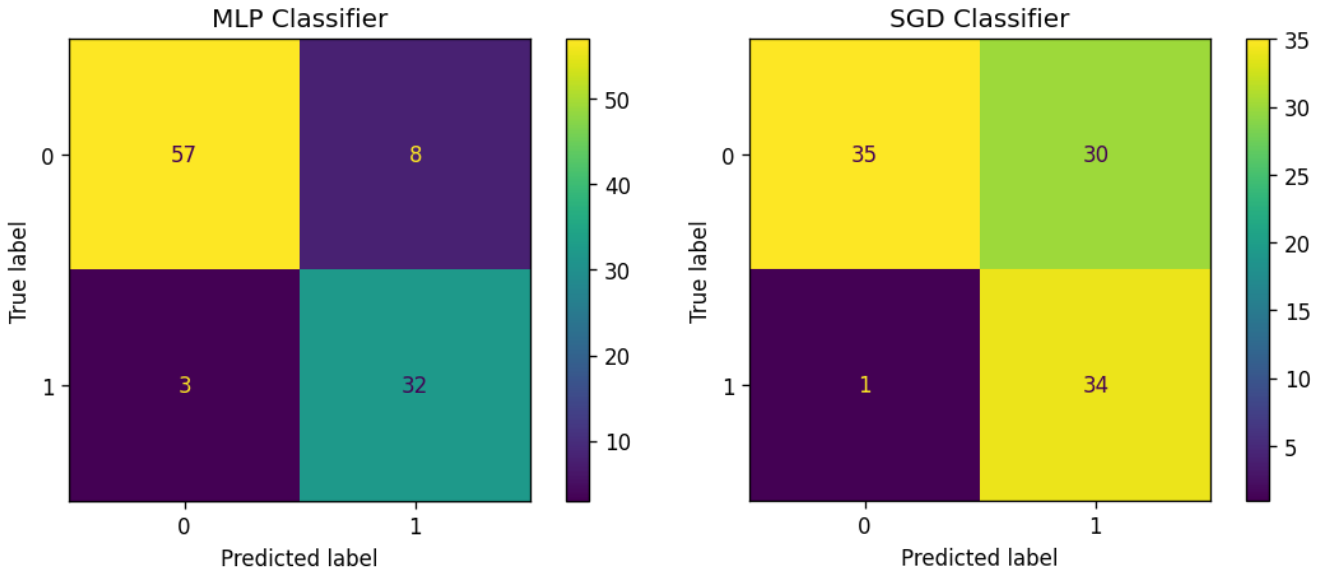
Figure 8. Training and validation accuracy plots for MLP and SGD models.

The training process results for the tuned MLP classifier model indicate that the model is learning because there is an increasing trend in the plot, the accuracy score increase as epochs increase. The training accuracy starts at about 33% and ends around 91%, and the validation accuracy starts at about 25% and ends around 90%. Both the training and validation curves in the tuned MLP classifier successfully converge which also indicates that the model is learning. However, the training process results for the default SGD classifier with default parameters do not show any sign that the model is learning, there is no trend in the curves, and they appear to be stationary, the accuracy scores for both training and validation keep fluctuating.

After the training process, both the trained tuned MLP and default SGD models are used to predict whether customers purchased a car or not, the predictions are used to generate the evaluation metrics discussed in section 3.3. Table 3 shows the values for the evaluation metrics for both the tuned MLP and default SGD models. Figure 9 represents the confusion matrices of both models that show the TP, TN, FP, and FN values from the generated predictions on the 100 testing samples.

Table 3: Summary of evaluation metrics values for MLP and SGD models

	CV Score	*Accuracy*	*Recall*	*Precision*	*Sensitivity*	*Specificity*
Tuned MLP model	0.9099	0.89	0.914	0.8	0.914	0.877
Default SGD model	0.8157	0.69	0.971	0.531	0.971	0.538



(a)

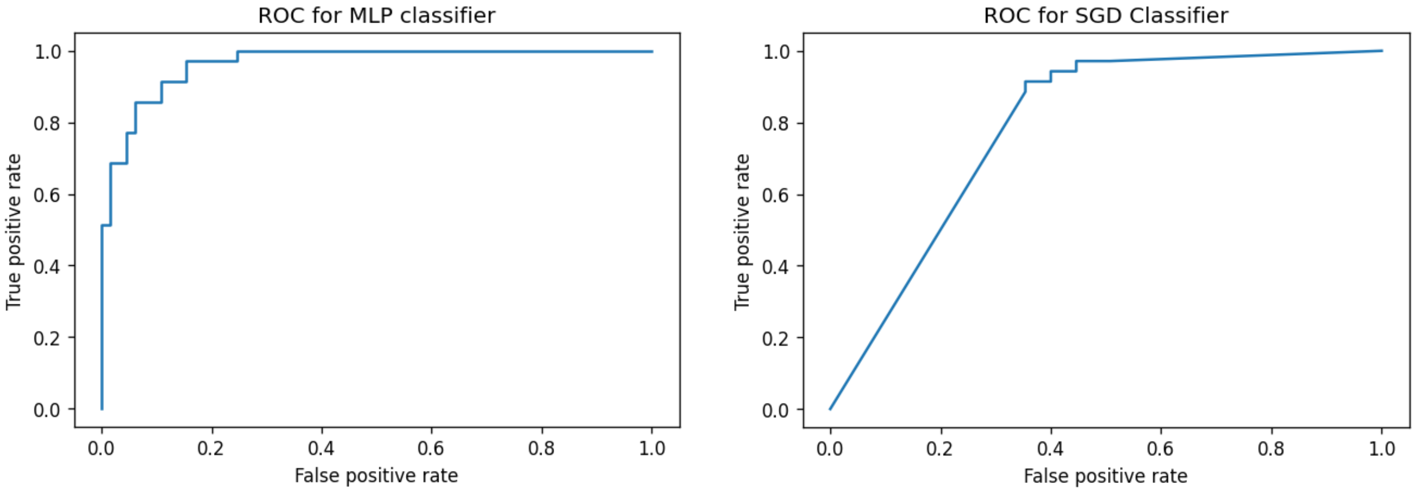
(b)

Figure 9. Confusion matrices for MLP and SGD models

The evaluation metrics clearly show that the tuned MLP model performs better than the default SGD model overall. The MLP model has an accuracy of 89% which is 20% higher than the accuracy of the SGD model which is 69%, this means that the tuned MLP model is more accurate overall. The precision value for the tuned MLP model is 80%, which is higher than the SGD model value which is 53.1%, this means that the quality of positive predictions made by the tuned MLP model is better than the SGD model. The specificity value for the tuned MLP model is 87.7%, which is higher than the SGD model value which is 53.8%, this means that the tuned MLP model has a higher ability to detect negative samples. However, the recall and sensitivity values for the MLP model are 91.4%, while the values for the SGD model are 97.1%, this means that the SGD model has a higher ability to detect positive samples which can be observed from the confusion matrix in Figure 9, where the SGD model only predicts one positive sample incorrectly out of 35 as opposed to the tuned MLP model which predicted three positive samples incorrectly out of 35.

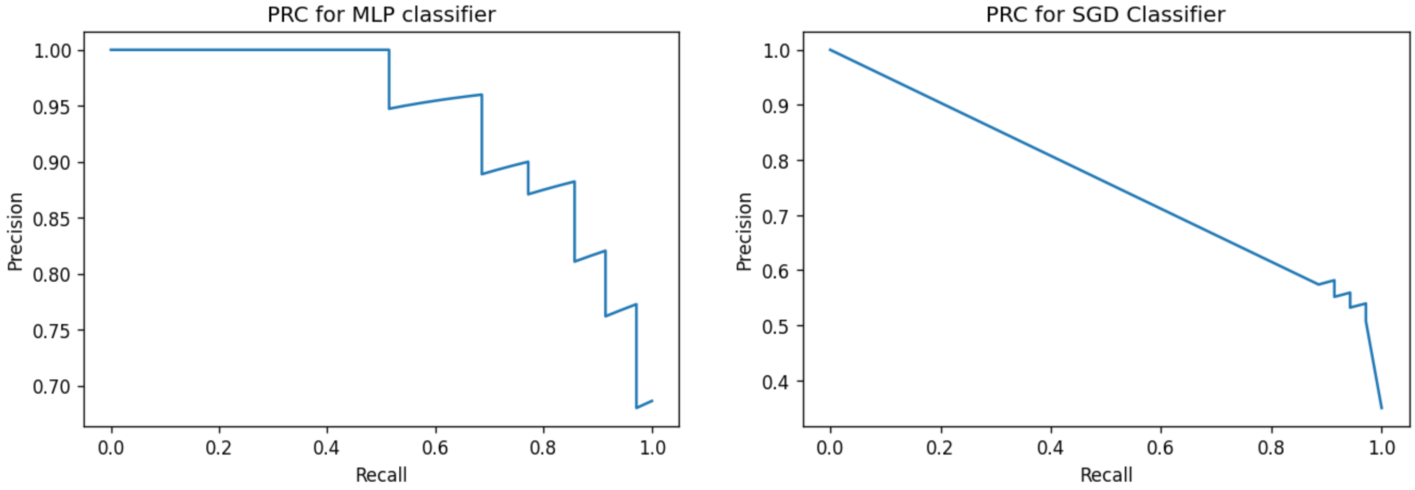
However, the result does not necessarily mean that the SGD model is better overall. Also, the tuned MLP model has a better cross-validation score than the SGD model, this indicates that the tuned MLP model is better for this problem.

The specificity and sensitivity metrics are used to generate the ROC curve while the precision and recall metrics are used to generate the PRC curve for both the tuned MLP and default SGD models. Figures 10 and 11 show the ROC and PRC curves respectively.



(a) (b)

Figure 10. ROC curves for MLP and SGD models



(a) (b)

Figure 11. PRC curves for MLP and SGD models

For the ROC curve, the more the curve converges to the top-left corner of the plot, the better the model is. From Figure 10, The ROC curve for the tuned MLP model converges more to the top-left corner of the plot than the default SGD model. For the PRC curve, the higher the area under the curve (AUC) is, the better the model will be. From Figure 11, the AUC for the PRC curve for the tuned MLP model is observed to be larger than the AUC for the PRC curve for the default SGD model. Both the ROC and PRC curves indicate that the tuned MLP model has better performance than the default SGD model.

In conclusion, the results show that the tuned MLP model is successfully learning at each training iteration while the default SGD model is not learning. The evaluation metrics values show that the tuned MLP model performs better overall than the default SGD model, except for the recall and sensitivity metrics which show that the SGD model has a higher ability to detect positive samples. Finally, the ROC and PRC curves for the tuned MLP model are better overall because the ROC curve converges to the top-left corner of the plot better than the SGD model and the PRC curve has a higher AUC than the SGD model. Therefore, the tuned MLP model is the best model for this problem and can better predict whether a customer purchases a car or not.

5. References

[1] Kaggle. (2022), “Cars – Purchase Decision Dataset”. Website, 2022. https://www.kaggle.com/datasets/gabrielsantello/cars-purchase-decision-dataset

[2] Popescu, M., Balas, V., Popescu, L., Mastorakis, N. (2009), “Multilayer Perceptron and Neural Networks”. *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*, Issue 7, Volume 8.

[3] Feng, J., He, X., Teng, Q., Ren, C., Chen, H., Li, Y. (2019), “Reconstruction of porous media from extremely limited information using conditional generative adversarial networks”. *PHYSICAL REVIEW E 100*, 033308.