MLDM 2021 Coursework Group: *Data Divas*

Bhavyasree Sasindra BS00916@SURREY.AC.UK
Hemalatha Raju HR00565@SURREY.AC.UK
Preethi Patrick PP00705@SURREY.AC.UK
Shailaja Janjirala SJ00933@SURREY.AC.UK
Swarnika Priyam SP01636@SURREY.AC.UK

Abstract

The project aim was to understand the application and evaluation of Machine Learning and Data Mining techniques. Three datasets, namely, Water Quality Classification, Mushroom Classification and Pyrimidine SAR (structure-activity relationship) are chosen for the analysis of the algorithms like Decision tree, Random Forest, Support vector machine, Naïve Bayes, Neural networks and k-means clustering. Accuracy and roc score are used as the performance metrics in the analysis of supervised learning algorithms and metrics like inertia and silhouette coefficient are used for K-means algorithm. Of these algorithms, random forest performs better for all three datasets. Out of the three datasets, performance of K-means is better on mushroom classification. Performance of inductive logic programming is better having good accuracy, but the learning and estimation time are higher. Mountain car problem has been chosen for analysis of q-learning algorithm in reinforcement learning and rewards. It is observed that as the epoch increases, the score is better, and the goal is achieved in minimum number of steps.

1. Project Definition

1.1 Water Quality Classification

The water quality dataset contains water quality metrics for 3276 different water bodies. Our objective was to classify water into potable and non-potable based on the given features. There are 9 features with two classes as potable and non-potable and 3277 data points. The features are pH value, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes and turbidity. Potability indicates whether the water is safe for human consumption or not. 1- means the water is potable and 0- means not potable. The dataset can be found at: https://www.kaggle.com/adityakadiwal/water-potability

1.2 Mushroom Classification

The Mushroom dataset contains information relating to 23 species of gilled mushrooms in the Agaricus and Lepiota Family of mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Our objective is to classify the mushrooms as edible or poisonous based on 23 features which are shape, surface and color of the cap, bruises, odor, attachment, spacing, size and color of gill, surface and color above and below the ring, shape and root of stalk, type and color of veil, type and number of rings, spore-print color, population and habitat. The dataset contains 8124 data points and is taken from:https://www.kaggle.com/uciml/mushroom-classification

2. Data Preparation

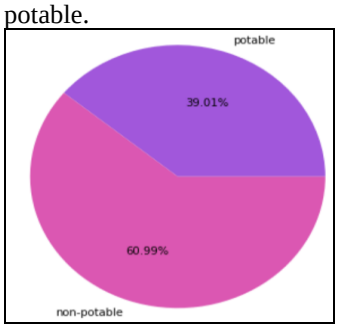
2.1 Water Quality Classification

2.1.1 DATA CLEANING

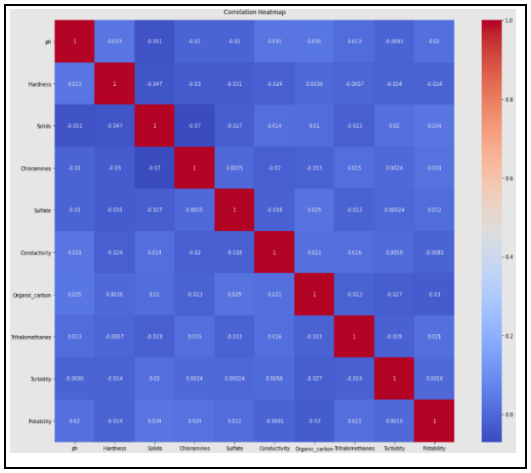
The dataset contains missing values for the following three features-pH, sulphate and trihalomethanes. As the dataset contained outliers, the missing values were replaced by median. The outliers were detected using box plot and were dropped later. The modellings were done on the new dataset which contains 3265 data points.

2.1.2 DATA EXPLORATION

The dataset contains 39.01% of data points belonging to class potable and 60.99% of data belonging to class non-potable.



There are no highly correlated features in the dataset. This was observed by plotting the below correlation heatmap.



2.2 Mushroom Classification

2.2.1 DATA CLEANING

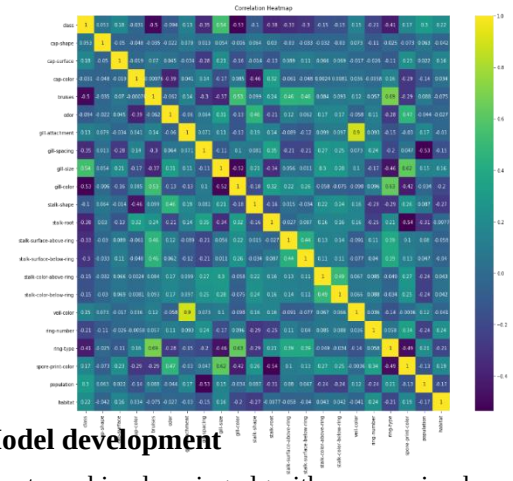
Upon exploration of the dataset, it was found that there were no null values and the dataset is balanced with 48.20% of data points belonging to poisonous class and remaining 51.80% belonging to edible class. The column 'veil-type' consisted of only single value i.e. p, so it was removed in order to prevent it from acting as a unique identifier. In order to observe the distribution of other features, their corresponding count plots were created. Various techniques of data exploration like heat map and correlation matrix were also done.

2.2.2 VARIABLE TRANSFORMATION

All the categorical features in the Mushroom dataset were converted to numerical format using label encoding. Those features with binary categories were converted using binary encoding. Whereas, others were encoded using the get dummies method for K-Means. These features were then normalized using scaling operation.

2.2.3 DATA EXPLORATION

The distribution of attributes and plots corresponding to their value counts were created. Correlation matrix and its heatmap were also plotted.



3. Model development

Different machine learning algorithms were implemented to analyze the water quality dataset and mushroom classification dataset. Supervised learning algorithms like Decision Tree, Random Forest, Support Vector Machine, Naïve Bayes, Perceptron, and Neural Networks were implemented. Unsupervised algorithm - K-means clustering was also implemented. Inductive Logic Programming and Reinforcement learning algorithms have been implemented.

3.1 Decision Tree, Gradient Boost and Random Forest

Decision Tree (DT), Gradient Boosting (GB) and Random Forest (RF) are machine learning methods used for both classification and regression tasks. For DT, the objective is to predict the value of a target by learning decision rules from the features. Though it is easy to understand, we have problems like overfitting, bias error, and variance error. RF is a collection of DTs and is a robust method used to overcome the problems of DTs. GB is also a collection of DTs but differs to RF. RF combines trees at the end and uses averages or majority rules, but GB combines trees at the beginning. Each tree is built independently for RF whereas one tree is built at a time for GB. Tuning parameters for GB can result in better performance than RF, but it may lead to overfitting if a lot of noise is present. RF performs well even if a lot of noise is present.

3.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for classification or regression tasks. The objective of SVM is to find a hyperplane in an N-dimensional space (N- number of features) that can classify the data points. A subset of the training examples called the support vectors are used to define the hyperplane.

It is possible to have many hyperplanes that can separate the two data classes. Our objective is to find a plane with maximum margin. Maximizing the data points provides generalization and avoid overfitting, as a result future data points can be classified with more confidence. SVM uses a set of mathematical functions called kernel. Kernel maps inputs from lower dimensions to higher dimensions. The kernel trick allows SVM to classify non-linear data by projecting it to a higher dimension where it is linearly separable by a hyperplane.

One notable advantage of using SVM over neural networks is guaranteed convex weight optimization. This means that there is no local optima, the solution is guaranteed to converge to one global optima. It is more effective in high dimensional spaces and is memory efficient.

SVM does not perform well when the dataset has noise i.e. when target classes overlap. SVM will underperform when the number of features exceeds the number of training examples.

3.3 Naïve Bayes

Naïve Bayes classification algorithm is based on Bayes theorem and is one of the simplest algorithms under supervised learning. In the Naïve Bayes classifier prior probability of the classes in dataset is calculated and the likelihood probability with every feature of each class is found. Using Bayesian formula, posterior probability is calculated to determine which class has higher probability. This algorithm is said to work at good speed on large datasets with high accuracy. The computational cost involved is also less. It can also be used to handle multi-class classification problems. The drawback of SVM is that, it assumes that there's no correlation between the features of the class which may sometimes lead to zero probability which makes the model unable to predict the class.

3.4 Neural Network

"Perceptrons" are one of the most basic classifiers. The performance of the perceptron will act as a baseline for further experiments done with MLP and DNN. Even though they are very simple they work exceptionally well

when the data is linearly separable. This means depending on the nature of the dataset, perceptron might even be a great choice. That is if the performance gain by introducing a DNN is marginal, the added computational complexity of the DNN cannot justify the minimal performance gain.

"Multi-Layer Perceptrons" are made of multiple Perceptrons. This is done to overcome the limitation of perceptron, i.e., the ability to classify datasets which are not linearly separable. This gives MLP the ability to work on more complex datasets. More often than not the problems in real world are not linearly separable. While a perceptron can be thought of as a line in the feature space trying to separate data points MLP is a set of line trying to do the same.

"DNNs(Densely Connected Neural networks)" are an evolution from MLP. While perceptron can be thought of as a single neuron, DNNs are formed by sequence of layers of neurons where each neuron in one layer is connected to all the neurons in the next layer. This opens up the possibility of working with far more complex datasets. They are also very versatile, meaning they can solve numeric, textual, image, audio, video datasets.

3.5 K-Means Clustering and PCA

The k-means clustering technique is an unsupervised data mining and machine learning tool that clusters observations into groups of related observations without knowing the links between them. The algorithm uses sampling to try to figure out which group, or categories, the data belongs to. The number of clusters to which the data belong is determined by the variable k.

The benefit of k-means clustering is that it tells you about the data (in its unsupervised form), rather than you needing to teach the algorithm about the data from the beginning (in its supervised version). Additionally, as all k-means does is compute the distances between points and group centers, it has the advantage of being quite quick. As a result, it has a linear complexity of O(n). However, K-Means also has some drawbacks. Firstly, it requires the user to input the number of clustering groups, rather than selecting the number by gaining insights from the data. Secondly, K-means starts with a random selection of cluster centers, therefore different runs of the method may provide different clustering results. As a result, the results may not be reproducible or consistent. To overcome this challenge "k-means++" was used for initializing centroids.

For the purpose of this coursework, K-Means was used in combination with PCA (Principle Component Analysis). PCA is a computing technique used for reducing dimensionality. It projects each data point onto only the first few principal components to produce lower-dimensional data while keeping as much variation as possible.

To determine the value of K (number of clusters) the elbow method (also known as the total within-cluster sum of squares) and Average Silhouette score were used. This was done only for the purpose of validation, because the number of clusters was already known for the chosen datasets. Metrics used for evaluating the performance of the three models are: Homogeneity score, Completeness score, V measure, Adjusted rand index, Adjusted mutual information & Silhouette coefficient.

3.6 Q-Learning

The foundation of reinforcement learning is laid by the entire paradigm of exploring the environment and learning through actions, rewards, and states. Reinforcement learning is used to handle problems involving sequential decision making and long-term goals, such as game playing, robotics, resource management, and logistics. Q learning algorithm helps to define main components of a reinforcement learning solution that is the agent environment action rewards and states. Q-learning is an algorithm that operates outside of the policy framework.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)]$$

We update our Q-function using Q-learning by presuming we're doing action a that optimizes our post-state Q function Q(st+1,a). We don't have any constraints on how the next action is chosen; we just have this "optimistic" viewpoint that all future action selections from every state should be optimal, so we choose the action a that maximizes Q(st+1,a). Given enough data samples, we can feed Q-learning data created by any behavior policy (expert, random, even poor rules) and it should learn the optimal Q-values.

4. Model evaluation / Experiments

The model evaluation and the experimental results of the supervised and unsupervised learning algorithms implemented with different datasets have been explained in this section.

4.1 Water Quality Classification

4.1.1 Null Hypothesis

Water quality is determined to check if it is potable or not. Potability indicates if water is safe for human consumption. Classes: Potable - 1 and Not potable - 0.

4.1.2 DECISION TREE, GRADIENT BOOST & RANDOM FOREST

A train-test split of 80-20 was used. DT, GB, and RF were implemented for the water potability problem.

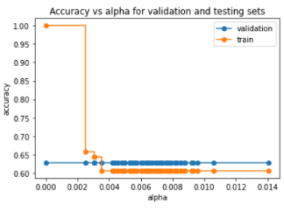
RESULTS & DISCUSSION

DT was visualized using plot_tree and later, graphviz for better visualization. Model was evaluated for train (accuracy - 100%) and test data (accuracy - 59.41%). To improve the accuracy, parameters were tuned. Test accuracy of **64%** was observed with a min_samples_split = 16, max_depth = 7. However, cost complexity pruning with a range of ccp_alpha values determined

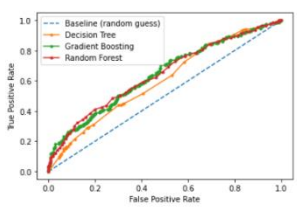
ccp_alpha=0.0022 to be the best, as seen in the first graph below which gave an accuracy of 62.78%. **GB** and **RF**, both gave an accuracy of **66.92%**. **ROC_AUC** score was observed as 0.603 for DT and 0.642 for GB and RF. Performing 10 folds stratified cross validation for these models gave an accuracy of 61.37%, 63.13%, and **65.54%** respectively for DT, GB, and **RF**. **Confusion Matrix** as shown below was also plotted.

Inference: Random Forest performs better amongst these 3 models with cross-validation. However, GB and DT also performs equally well.

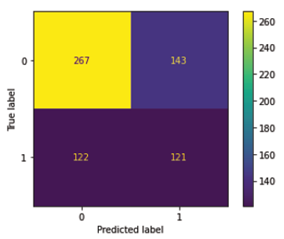
Pruning Decision Trees with cost complexity



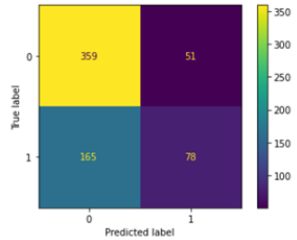
ROC Curve for DT, GB, RF



Decision Tree



Random Forest



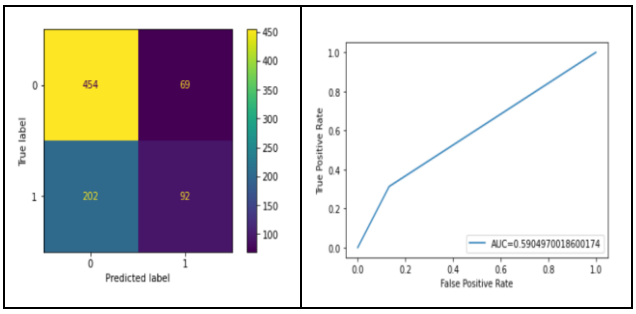
4.1.3 SUPPORT VECTOR MACHINE

A train- test split of 80-20 was used. GridSearch was conducted for parameter tuning. The following parameters were tuned- C, gamma, and kernel. C indicated the regularization parameter. It trades off correct classification of training example against maximization of decision margin. Gamma defines the influence of a single training instance and kernel describes the function to be used.

RESULTS & DISCUSSION

An accuracy of 0.66 was obtained initially. After fine tuning, the best parameters were observed to be rbf kernel with gamma 0,1 and C=4. The accuracy slightly improved to 67% after fine tuning. The ROC_AUC score is observed as 0.59049700186.

Figure shows the confusion matrix and roc curve after fine tuning.

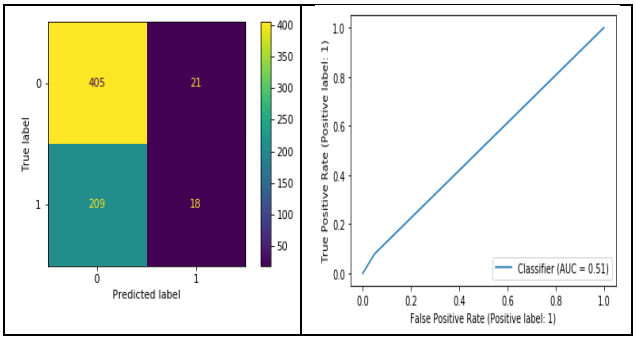


4.1.4 NAÏVE BAYES

Gaussian Naïve Bayes classification algorithm has been implemented. The prior probabilities of the classes and the variance of the features may be included while defining the model, if only they need to be adjusted for the given data. In this case, no such prior probabilities are used.

RESULTS & DISCUSSION

The accuracy score and the mean absolute error predicted by the model are observed as 0.63 and 0.37 respectively. The ROC_AUC score is observed as 0.55. When experimented using Stratified Kfold for 10 folds, the accuracy has improved and is observed to be 0.65 and the mean absolute error is 0.35. Using the stratified kfold, the performance of the Gaussian Naïve Bayes classifier has improved. Figure shows the plots of confusion matrix and roc curve for 10 folds experiment.



4.1.5 NEURAL NETWORK

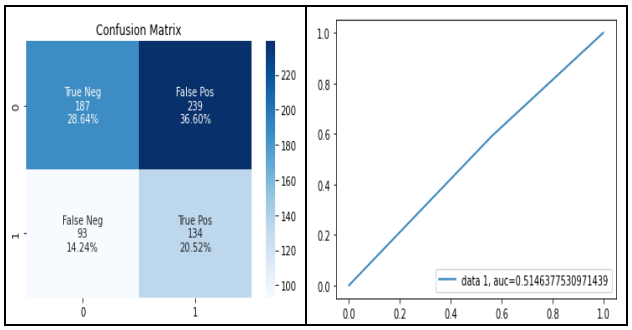
Perceptron was trained by tuning learning rate and number of iterations to obtain optimal results.

The **Multi-Layer Perceptron** was trained by tuning activation function, hidden layer sizes and number of iterations to obtain optimal results. **DNN** was also implemented.

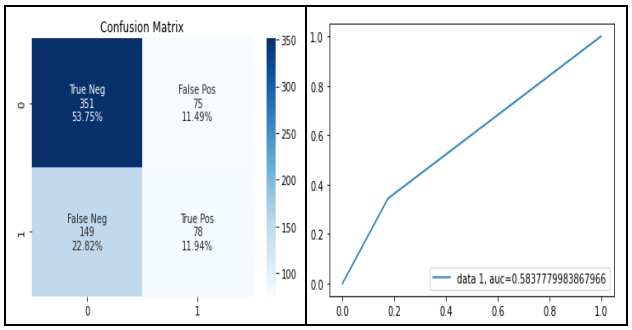
RESULTS & DISCUSSION

The **Perceptron** was trained with learning rate set to 0.1 and iterations set to 400 to get an AUC score of 0.514. The same was tried with learning rate set to 0.01 and 0.001, both yielding slightly worst results. Turning up the iterations to 1000, 10000 also didn't improve the result. Furthermore, the model was trained with 100, 200, 300 iterations, each of which yielded slightly worse results. This process is how the parameters were decided. From the confusion matrix it can be seen that the model leans

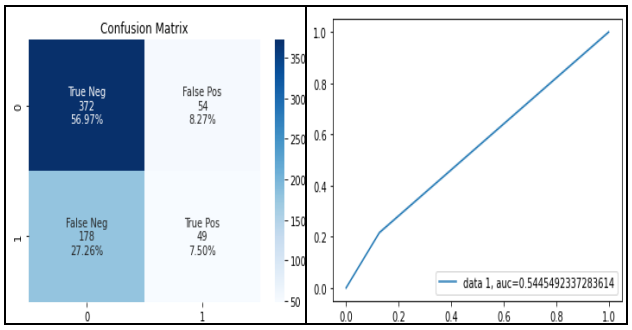
towards predicting the positive class more than the negative.



The **MLP** result was obtained using activation function as ReLU, a single hidden layer with 20 neurons and 1000 iterations to get an AUC score of 0.584. The model was trained by setting up 2 hidden layers with 20 and 16 neurons respectively, which yielded worse performance. It was again retrained by bringing down the number of neurons in the single hidden layer to 8 neurons which again brought down the performance. These experiments were conducted with 400, 1000, 10000 iterations , out of which 1000 out performed in all cases.



The DNN result was obtained with 2 hidden layers with 8 neurons each. Both hidden layers uses ReLU as the activation function while the output layer uses sigmoid. The model was trained with kfold approach using 10 folds. After experimenting with 16, 128 and 512 for batch size, 128 was finally chosen. While 512 resulted in a much worse case of overfitting, 16 ended up underfitting the model.



The trained model gave an AUC score of 0.54. From the confusion matrix it's clear that the model is skewed towards predicting the negative class.

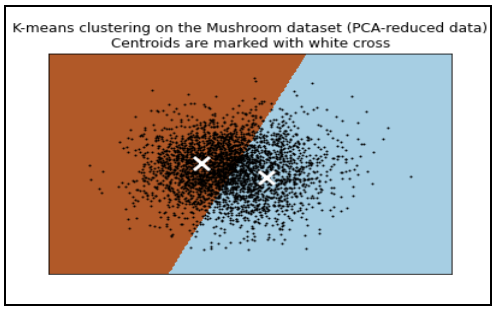
4.1.6 K-MEANS CLUSTERING

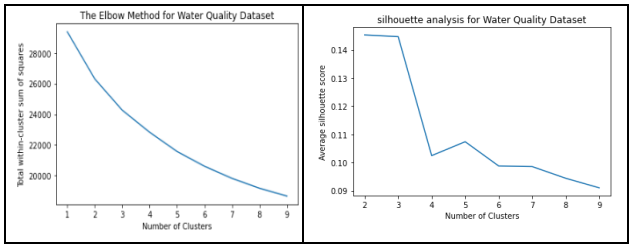
The 10 numerical features of the pre-processed water quality data were scaled, and K-means clustering was performed. "n_clusters" value was set to number of unique values of the dependent variable (2), "n_init" was set to 10, "sample_size" to 300 and other parameters to their default values. PCA was used to check the performance before and after dimensionality reduction

RESULTS & DISCUSSION

```
-------------- Metrics before PCA -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     9       0.13s   26311   0.000   0.000   0.000   -0.003  -0.000  0.136
-------------- Metrics after PCA with r=10 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     8       0.12s   23678   0.000   0.000   0.000   0.001   -0.000  0.137
-------------- Metrics after PCA with r=4 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     4       0.13s   11632   0.000   0.000   0.000   0.001   0.000   0.178
-------------- Metrics after PCA with r=2 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     2       0.11s   4839    0.000   0.000   0.000   -0.001  -0.000  0.305
```

The value of K, number of clusters is validated using the elbow method and silhouette analysis. A bend in the elbow curve at k value of 2 is observed. Similarly, the silhouette clusters for k value 2 are uniform and separated.





4.2 Mushroom Classification

4.2.1 NULL HYPOTHESIS

The dataset with features of mushroom has been selected. Various classification algorithms are applied to classify if the mushroom is safe to eat or is deadly poisonous. Classes: Edible - e, Poisonous – p.

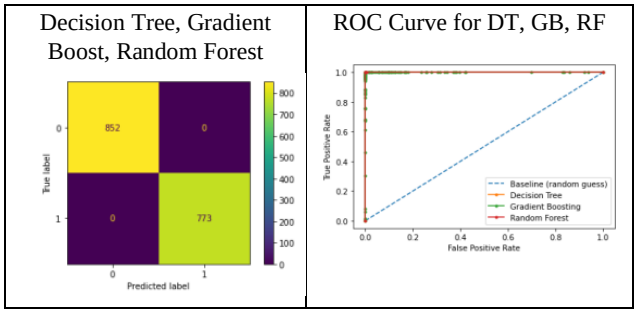
4.2.2 DECISION TREE, GRADIENT BOOST & RANDOM FOREST

A train-test split of 80-20 was used. DT, GB, and RF were implemented for the Mushroom Classification problem.

RESULTS & DISCUSSION

DT was visualized using plot_tree. It was evaluated for train and test data. **GB** and **RF** were also implemented.

All three classifiers gave the same result, accuracy being **100%** in all the cases. Therefore, no parameters were tuned in this case. Performing 10 folds stratified cross validation for these models also gave an accuracy of **100%**. The **confusion matrix** and **ROC** (ROC_AUC = 1.000) obtained in each case had the same result as expected. Hence, the figures below show the same for each model.

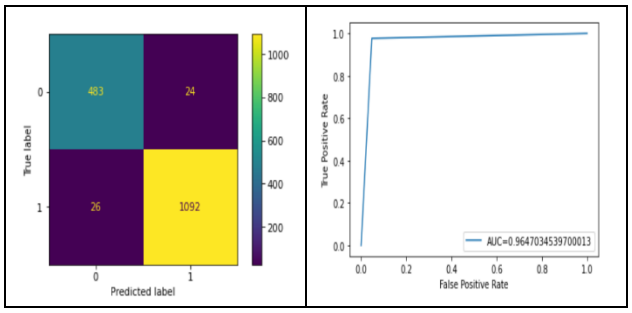


4.2.3 SVM

Classification of mushroom is a binary classification problem. SVM can be used for this purpose. SVM would construct a hyperplane separating edible ones from non-edible ones.

RESULTS & DISCUSSION

Initially an accuracy of 0.9267 was observed when using stratified kfold with 10 fold. On fine tuning the parameters with C=2, gamma=0.1 and kernel=rbf, the accuracy was improved to 0.969230. The final AUC score obtained is 0.9647. The figure below shows the confusion matrix and AUC score.

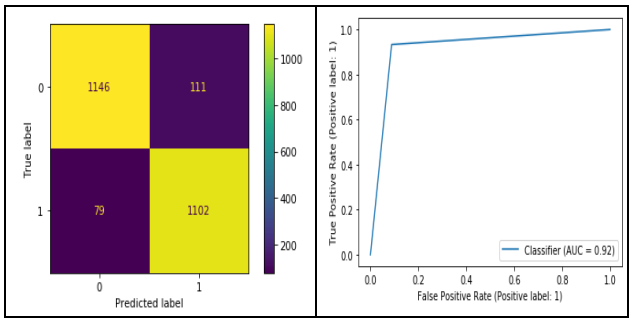


4.2.4 NAÏVE BAYES

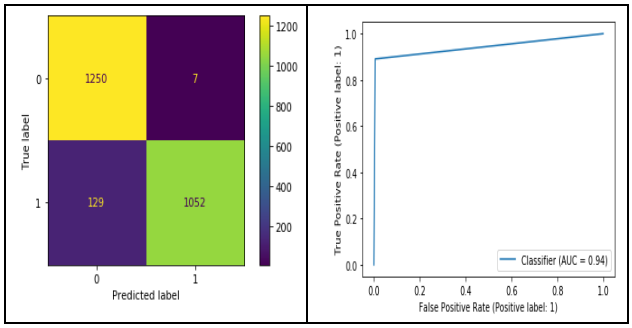
Gaussian Naïve Bayes and Categorical Naïve Bayes classification algorithms have been implemented. The features and classes are categorical and hence categorical NB classifier is used to obtain better classification.

RESULTS & DISCUSSION

Both Gaussian NB and Categorical NB were implemented for single run and for 10 fold using Stratified Kfold. Gaussian NB has obtained an accuracy of 0.92 and categorical NB has achieved an accuracy of 0.94 and the mean absolute error is observed as 0.078 and 0.056 respectively. Hence, it is observed that categorical NB classifies better. The ROC_AUC_Score observed for 10 fold in Gaussian NB classifier is 0.92 and Categorical NB classifier is 0.94. The roc-auc score is also found to be better in Categorical NB classifier. The observed roc curve and confusion matrix plots of Gaussian NB and categorical NB for 10 folds are shown in figures.



Confusion Matrix & ROC curve for 10 fold Gaussian NB model

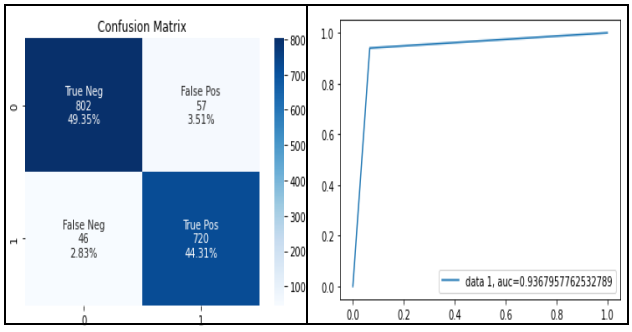


Confusion Matrix & ROC curve for 10 fold Categorical NB model

4.2.5 NEURAL NETWORK

RESULTS & DISCUSSION

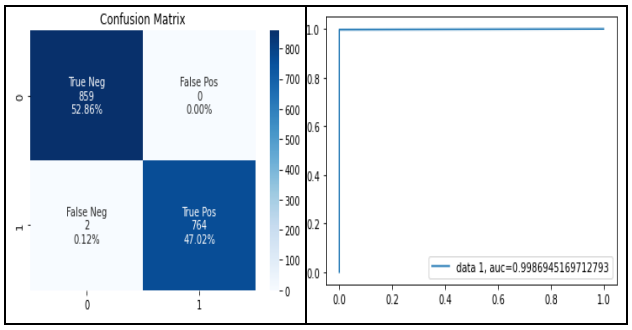
The Perceptron was trained with the learning rate set to 0.001 and maximum iteration to 400. With learning rate set to 0.01 & 0.1 slightly worse performance was observed and the learning rate was reverted to 0.001. Turning up the number of iterations however did not produce big improvements and 400 was observed as a nice middle ground.



The AUC score obtained with the parameter tuned is 0.936. The confusion matrix also a nice balance is the True Positives and True Negatives.

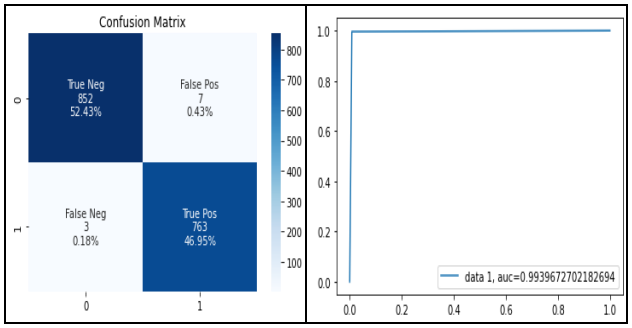
The MLP was trained with a single hidden layer with 4 neurons, activated using ReLU and using 400 iterations. Initially the number of neurons were 20 and the accuracy was 1, with that the model was retrained with 8 neuron

which produced accuracy 0.99. This lead to another training with 4 neurons, this led to a significant drop in the accuracy to 0.97. At this point the activation function was logistic, changing that to ReLU gave an accuracy of 0.99. This process is what lead to the choice of parameters for MLP.



The MLP finally with the chosen parameters produced and AUC score of 0.998. The confusion matrix also shows exceptional performance

The same DNN architecture was used in this dataset as well. The model was trained with kfold approach with 10 folds with 128 as batch size and 100 epochs each. Adding more layers or neurons did not have a huge impact in the performance which led to the 2 layer 8 neuron architecture. Increasing the epochs also led to only a minor improvement which didn't justify the increased training time. The batch size was set as 128 after testing 16 and 512, while 16 didn't help the model learn much 512 made the model overfit.



With the above mentioned parameters the DNN model was able to achieve an AUC score of 0.993. The confusion matrix also presents a very positive case for the DNN model.

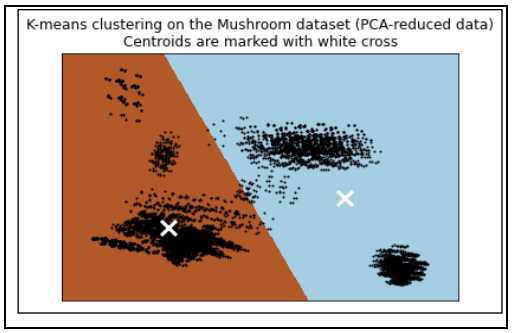
4.2.6 K-MEANS

The dataset containing 111 features after numerical encoding were reduced using PCA to 48, 24 and 2 and their respective performance is reported.

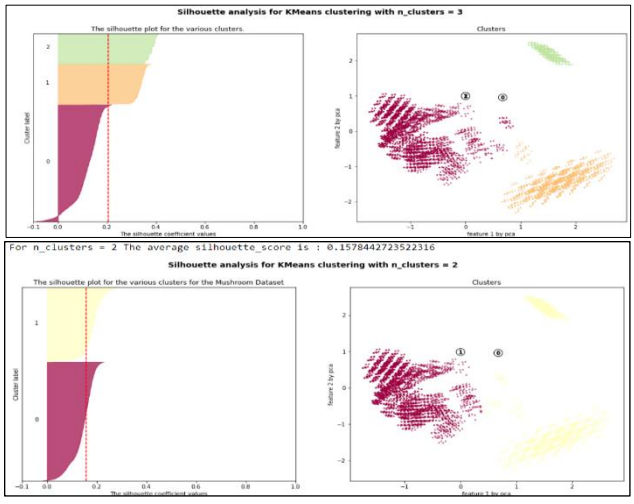
Parameters are, "n_clusers" set to number of unique values of the dependent feature, that is 2, "n_init" set to default value 10 and "max_iter" to 300

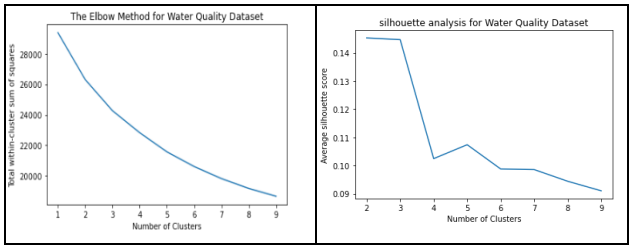
RESULTS & DISCUSSION

It can be observed that there was a significant improvement in performance metrics for PCA reduced dimension to 2. Especially, in the silhouette coefficient.



The results of the elbow method and silhouette analysis validated the chosen value for K as 2. The figure below (for K =2) indicates a homogenous distribution of clusters a negligible negative silhouette score in comparison to the distributions for K values ranging from 3 to 10. Also, the bend elbow curve is also seen around the same value.





4.3 Inductive Logic Programming

ILP helps in constructing easy-to-understand logical rules for SAR (Structure Activity Relationship) problems, which is used in the pharmaceutical R&D to find out improved variants of patented active drugs. The team has considered the Pyrimidine Problem for the inhibition of E. Coli Dihydrofolate Reductase. The main aim is to build a predictive theory relating the structure of a compound to its activity which can be used to select the structures with

high or low activity of untried drugs. Learning about these relationships, one can devise clinically effective, non-toxic drugs.

4.3.1 NULL HYPOTHESIS

The perl file of inductive logic programming (with background knowledge) uses aleph to build a predictive theory relating the structure of a compound to its activity.

4.3.2 ILP

Single perl file (Pyrimidines_B1.pl) was created using the background file with ".b" suffix. All the mode declarations which define the hypothesis language were written very carefully. The names of positive example files with ".f" suffix and negative examples with ".n" suffix was changed to "split1" to "split10" for 10-fold cross-validation. This "splitN" has been used in the utils.py file (used from Lab 7 – Surrey Learn) from where everything is imported in the main code. The positive and negative example files were used to create a Pyrimidines_Num.csv file. The accuracies and learning time for different algorithms have also been plotted. Pyrimidines_B0.pl was tested with positive and negative examples (from split1 files) before proceeding for Pyrimidines_B1.pl. The data was taken from https://www.doc.ic.ac.uk/~shm/e_coli.html

RESULTS & DISCUSSION

Sample Output from 5th & 6th fold

100% accuracy was observed for the 6th fold of ILP which gave a set of structural rules for the model. A couple of sample rules from 6th fold are discussed below.

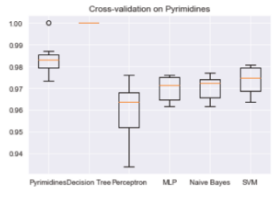
Sample Rule1: 'great(A,B) :-struc(B,C,D,C), flex(D,flex7), struc(A,E,F,C)' which is a positive example of a pair of drug states that E. Coli Dihydrofolate Reductase inhibition by drug A is higher than that by B and the chemical structure represents that drug B has an atom/group C substituted at position 3, an atom/group D substituted at position 4, and an atom/group C at position 5 with D having a flex of value 7. Also, drug A has an atom/group E substituted at position 3, an atom/group F substituted at position 4, and an atom/group C at position 5.

Sample Rule2: 'great(d10,d02)', 'great(d43,d02)' are learnt directly which are positive examples of a pair of drug and states that E. Coli Dihydrofolate Reductase inhibition by drug d10 is higher or having a greater activity than drug d02 and that drug d43 is higher or having a greater activity than drug d02 respectively.

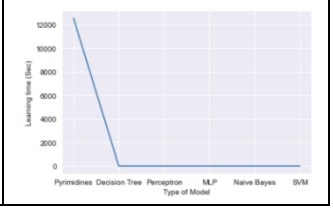
Table below presents the evaluation time (in secs) and accuracy for ILP and other non-relational models.

ILP	Learning Time	Evaluation Time	Accuracy	with std of
	12541.8	12578.04362	0.983256	0.006903
		Evaluation Time	Accuracy	with std of
Decision Tree	Train data	0.007229	1.000000	0.000000
	Test Data	0.010874	0.992539	0.004098
Perceptron	Train data	0.007283	0.960405	0.012253
	Test Data	0.007677	0.952077	0.026490
MLP	Train data	1.602115	0.969966	0.005484
	Test Data	0.812629	0.973604	0.009045
Naïve Bayes	Train data	0.005118	0.970158	0.005220
	Test Data	0.005809	0.974177	0.008400
SVM	Train data	0.252339	0.973792	0.005696
	Test Data	0.055052	0.974177	0.009675

Box Plot for Accuracy



Learning Time Plot



Inference: As evident from the table and plots above, ILP takes the longest time followed by MLP amongst other non-relational models for test data to learn the model for 10 folds cross-validation.

Decision Tree has the highest average accuracy both for train (100 %) and test data (99.25%) followed by **ILP** (98.33 %). The other models are very fast in comparison to ILP.

4.3.3 DECISION TREE, GRADIENT BOOST & RANDOM FOREST

A train-test split of 75-25 was used. DT, GB, and RF were implemented for the Pyrimidine (SAR) problem.

RESULTS & DISCUSSION

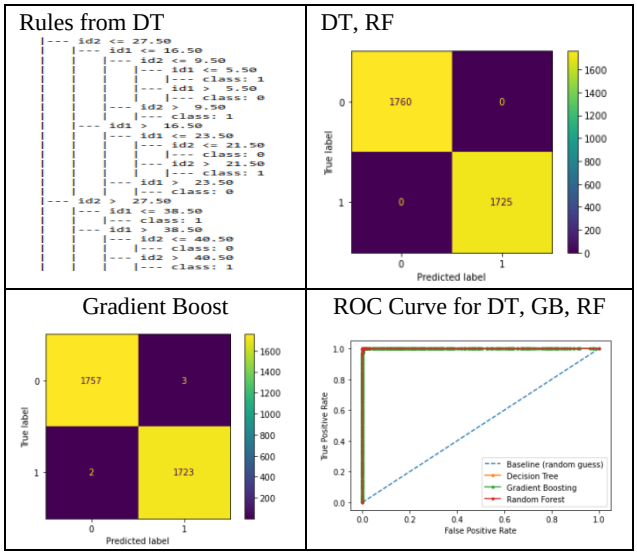
Rules for **DT** were extracted and can be read like:

IF (id2 <= 27.50) AND (id1 <= 16.50) AND (id2 <= 9.50) AND (id1 <= 5.50) THEN class=1

IF (id2 <= 27.50) AND (id1 <= 16.50) AND (id2 <= 9.50) AND (id1 > 5.50) THEN class=0

The set of rules can further be reduced depending upon the conditions. **DT** was visualized using plot_tree and later, graphviz for better visualization. It was evaluated for train and test data (100% accuracy for both). **GB** (99.85% accuracy) and **RF** (100% accuracy) were also implemented. The classifiers performed well, and no parameters were tuned. Performing 10 folds stratified cross validation for these models also gave an accuracy of **100%** for **DT** and **RF** and **99.69%** for **GB**. The **confusion matrix**

and **ROC** (ROC_AUC = 1.000) obtained in each case had the same result as shown below.

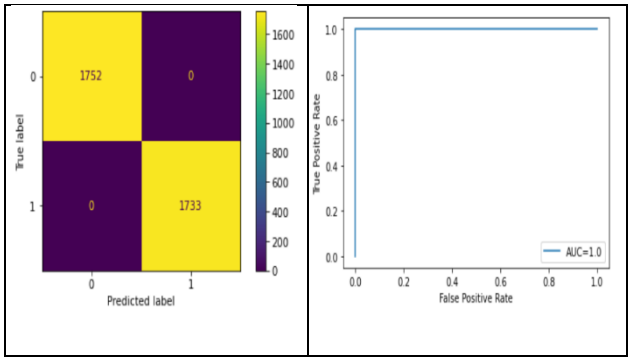


4.3.4 SVM

SVM was implemented using stratified kfold with 10 fold cross validation. Parameter tuning was also done to find the best parameters.

RESULTS & DISCUSSION

SVM gave an accuracy of 0.98 initially. On fine tuning the parameters using GridSearch with C=1, gamma=1 and kernel=rbf, an accuracy of 1.0 was obtained. The corresponding AUC score also improved to 1.0. The figure below shows the confusion matrix and ROC curve.

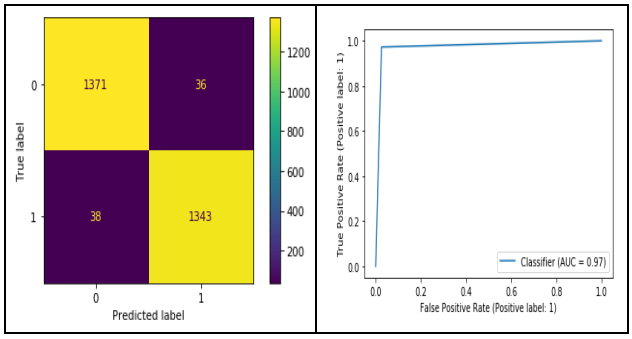


4.3.5 NAÏVE BAYES

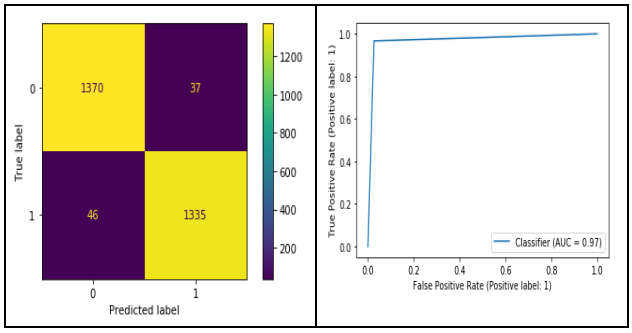
The dataset has categorical data and hence both Gaussian Naïve Bayes and Categorical Naïve Bayes classification algorithms have been implemented.

RESULTS & DISCUSSION

The accuracy results obtained using Gaussian NB and Categorical NB models for 10 folds are 0.9735 and 0.9702 respectively. The accuracies of both the algorithms are almost the same. However, the accuracy of Gaussian NB is found to be better. Mean absolute error observed for Gaussian NB and Categorical NB classifiers are 0.027 and 0.03 respectively. Figure shows the plots of confusion matrix and roc curve of Gaussian NB and Categorical NB models.



Confusion Matrix & ROC curve for 10 fold Gaussian NB model

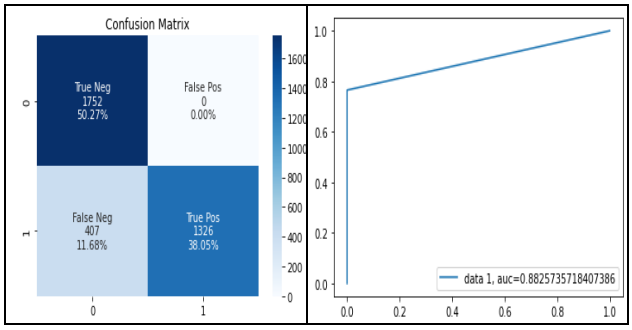


Confusion Matrix & ROC curve for 10 fold Categorical NB model

4.3.6 NEURAL NETWORK

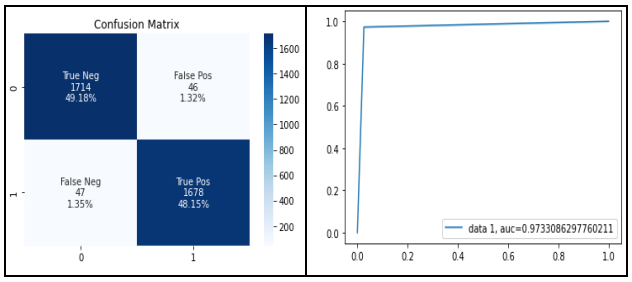
RESULTS & DISCUSSION

The **perceptron** was trained with a learning rate of 0.1 and iterations set to 400. Changing the learning rate to 0.01 & 0.001 resulted in the same performance of around 0.88 accuracy. The number of iterations when turned up to 1000 only slightly improved the performance which didn't justify the jump up in the training time.



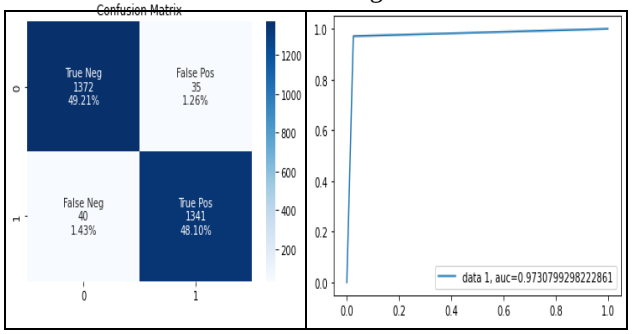
The perceptron finally gave an AUC score of 0.88. The confusion matrix shows good balance between True Positives and True Negatives.

The **MLP** model was designed with a single layer with 2 neurons using activation function ReLU and iterated for 100 times. Turning up the number of neuron and number of iterations has barely managed to affect the accuracy, the accuracy changed from 0.973 to 0.974.



The MLP model finally produced an AUC score of 0.973 and confusion matrix also presents a very positive case for the MLP model.

The **DNN** model was trained with the k fold approach with 10 folds with each fold running for 100 epochs with a batch size 128. It was observed that turning up the epoch did not meaningfully improve the performance of the model. The batch size of 16 and 512 also was not ideal as the batch size 16 resulted in a measurably worse performance while a batch size of 512 caused overfitting.



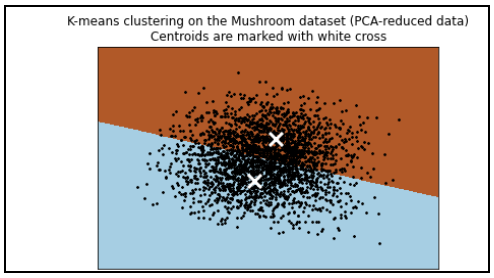
The DNN model finally gave an AUC score of 0.973. From the confusion matrix it can be seen that the model has a good balance between the True Positives and the True Negatives.

4.3.7 K-MEANS

The ILP Dataset (Pyrimidines) consists of 2 independent (categorical) and 1 dependent features. As K-means needs numerical encoding was performed on the categorical fields resulting in 110 attributes. These attributes were scaled for normalization. Parameters, n_clusters set to 2 (validated using elbow method and silhouette analysis) and rest were left as default. PCA was used with r value 42, 24 and 2 and corresponding performance is reported.

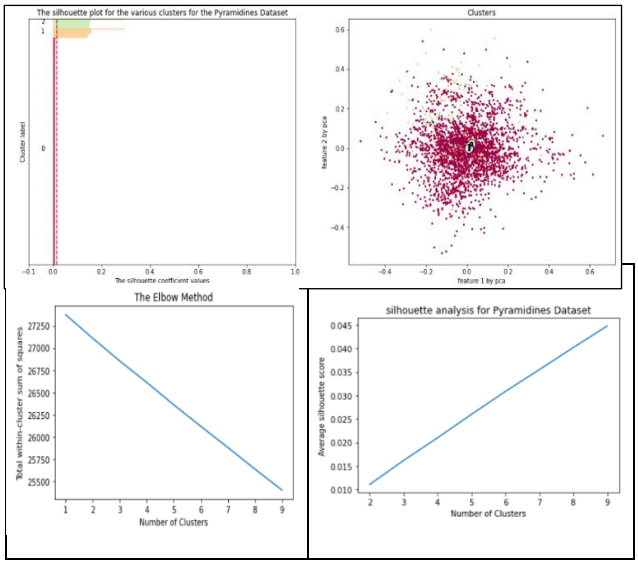
RESULTS & DISCUSSION

```
n_types: 2,      n_samples 13940,       n_features 110
-------------- Metrics before PCA -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     110     0.27s   1519180 0.003   0.024   0.006   0.000   0.006   0.010
-------------- Metrics after PCA with r=48 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     48      0.30s   679373  0.001   0.001   0.001   0.000   0.001   0.015
-------------- Metrics after PCA with r=24 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     24      0.34s   336059  0.009   0.010   0.009   0.012   0.009   0.033
-------------- Metrics after PCA with r=2 -------------------
init          feat    time    inertia homo    compl   v-meas  ARI     AMI     silhouette
k-means++     2       0.18s   20201   0.027   0.027   0.027   0.037   0.027   0.319
```



It is evident from the above figure that the K means algorithm is not suitable for performing clustering on this dataset. The clusters are neither homogenous nor distinguishable. That is because, the dataset consisted of purely ordinal data where the importance of one attribute over another is not clear. The integral part of k means clustering is to find minimal distance between various data points which when applied to categorical data becomes challenging. Even though the data is encoded into numerical values, there is no weightage assigned to different attributes rendering it difficult to distinguish between them. A way around this problem is to assign weights to different attributes during label encoding. However, doing that required domain expertise and knowledge about the activity of Pyrimidines in response to various combinations.

Silhouette Analysis and elbow plot also reflect the same. The distribution of clusters is not clear, and centroids are close to each other.

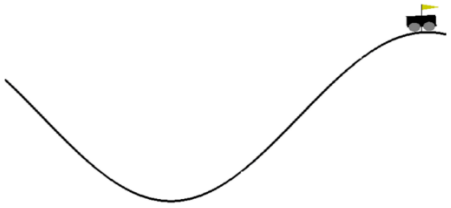


4.4 Q-learning

4.4.1 NULL HYPOTHESIS 1

Between two mountains, a mountain car is on a one-dimensional track. The objective is to drive up the right side of the mountain (reaching the flag). The car's engine, on the other hand, isn't powerful enough to ascend the mountain in one go. As a result, the only way to succeed is to maintain momentum by driving back and forth.

4.4.2 MATERIAL & METHODS



The Mountain Car environment is implemented by the OpenAI Gym library. We have defined the position space and velocity space. Action space is defined as 0, 1 and 2 with 0 meaning accelerate car to the left, 1 - no action and 2- accelerate car to the right. Two different reward functions have been implemented to learn the algorithm. Equation gives the Q learning equation implemented in the algorithm.

Q[state, action] = Q[state, action] + alpha*(reward + gamma*Q[state_, action_] - Q[state, action])

RESULTS & DISCUSSION

We initially set epsilon to 1.0. This gradually decreases over time to a finite value. We have set the episodes steps to 1000 because the initial 200 steps defined by default would not be sufficient to achieve good results. A reward of 0 is awarded if the agent reaches the flag (position=0.5), a reward of -1 is awarded if the position of the agent is less than 0.5.

The algorithm was also experimented for different learning rates (alpha) and epochs. Figure shows the score against the epochs for 50000 epochs with learning rate of 0.01. Best score is found to be -0.84.

From the figure, it is evident that the reward keeps on increasing over epochs, this indicates that the agent is learning. For each epoch the agent tries to maximize its reward.

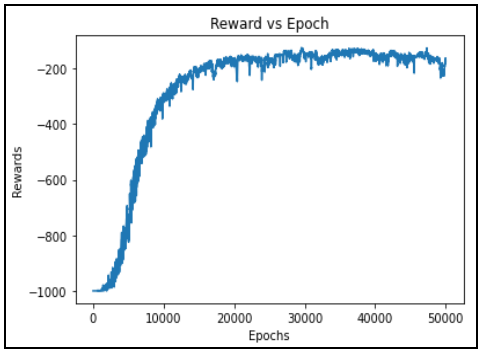
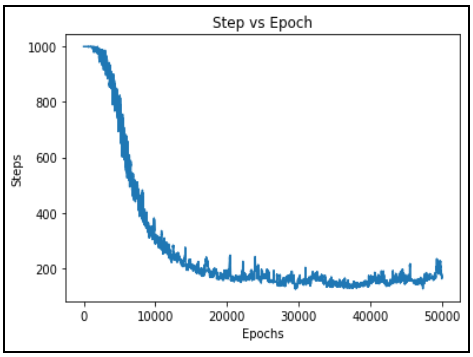
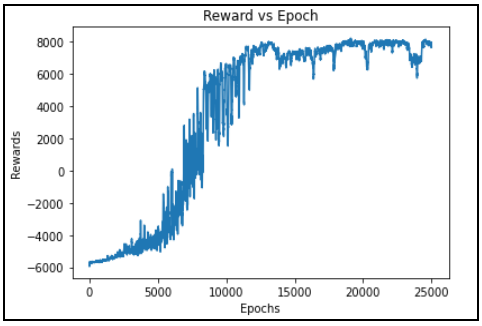


Figure Score Vs epochs for 50000 epochs with 0.01 lr

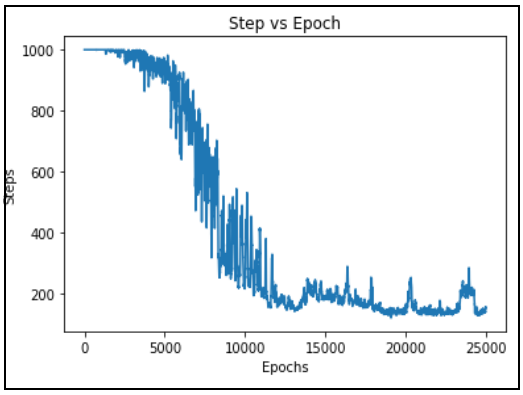
On increasing the episodes, the algorithm took fewer steps to learn. The figure shows that when number of epochs increases the algorithm is able to learn faster and more efficient ways to accomplish the goal.



Another experiment was to change the reward function. Here if the current state is higher on either sides of the mountain, then then a +20 reward is given else the reward is -25. If the goal is reached, then the reward is 10000.



In the above figure we can see reward increasing as the number of epochs increases and also in the figure below we see a similar trend after changing the reward function. But something to note is the difference in the smoothness of the curve which indicates the initial reward function generates a more stable model.



5. Discussion of the results

Dataset	Algorithm	AUC Score	Accuracy
Water Quality	DT / GB / RF	0.603/0.642/0.642	0.64 /0.66 /0.66
	Gaussian Naïve Bayes	0.55	0.63
	SVM	0.590	0.66

	Perceptron/ MLP/ DNN	0.514, 0.584, 0.54	0.49,0.66,0.64
Mushroom Classification	DT / GB / RF	1.0 / 1.0 /1.0	1.0 /1.0 /1.0
	Naïve Bayes (Gaussian, Categorical)	0.92, 0.94	0.92, 0.94
	SVM	0.9647	0.97
	Perceptron/ MLP/ DNN	0.936, 0.998, 0.973	0.94,1.0, 1.0
Pyrimidine SAR	DT /GB / RF	1.0 /1.0 /1.0	1.0 /99.85 /1.0
	Naïve Bayes (Gaussian, Categorical)	0.97, 0.97	0.9735, 0.9702
	SVM	1.0	1.0
	Perceptron/ MLP/ DNN	0.88, 0.973, 0.973	0.88,0.97, 0.97

In water quality we observe that among all the models Random Forest performs the best. The Random Forest will outperform a large number of reasonably uncorrelated models (trees) working as a committee. The key is the low correlation between models. We can see a similar performance in the mushroom classification dataset for Random Forest. Also, all the other models are performing well, this is because the model assumptions matched the characteristics of the data. In Pyrimidines SAR dataset, created from the perl file, Random Forest and SVM shows exceptional performance. This can be because there are only two features.

In the case reinforcement learning, we experimented with Qlearning algorithm on Mountain car environment. We observed that for the both the reward functions, as the number of epochs increases the learning becomes better and the number of steps taken to reach the goal is minimum.

Unsupervised learning K-Means algorithm performed best on the Mushroom classification dataset, resulting in clusters that are clearly separated and homogenous (low inertia). It showed poor performance on Water Quality classification dataset mainly due to its small size. K-means is known to perform well on very large datasets, as a result the clusters were not clearly distinguishable.

K-Means Metrics	Inertia	Homo genity	Complete -ness	v-means	Silhouette coeff
Water Quality	4839	0.0	0.0	0.0	0.305
Mushroom	78248	0.573	0.596	0.584	0.589
Pyrimidines SAR	20201	0.027	0.027	0.027	0.319

6. Conclusions

From the experiments, it was observed that Random Forest outperforms other models. One advantage of using RF is that it works well with categorical and continuous variables and is less prone to overfitting. The models presented here are analyzed on synthetic data, hence their performance might be limited on real-world data. For the Pyrimidines dataset, we get high accuracy due to limited features, including more background knowledge would give a much more reliable model. K-means performs well on large datasets and thus gave good results for mushroom dataset. For the Pyrimidines dataset, though it was large, its ordinal attributes could not be weighted appropriately while numerical encoding, making it difficult to classify. The accuracy achieved using ILP is better, but the learning and evaluation time are high. The QL algorithms learn better over time and takes minimum number of steps to achieve the goal. Discretization of the continuous state space for QL will lead to inefficient learning, due to curse of dimensionality. Deep QL can solve this problem.

Contributions

Bhavyasree (6660767)- Data preparation and data visualization for water quality and mushroom dataset. Implemented SVM for all 3 datasets, contributed to implement q-learning. Contributed to the following sections for the report- Abstract, Section 1, 2, 3.2, 3.6, 4.1.3, 4.2.3, 4.3.4, 4.4, 5,6.

Hemalatha (6671273) – Data preparation and data visualization for water quality and mushroom dataset. Implemented Naïve Bayes for all 3 datasets and contributed to implement Q-learning algorithm in Reinforcement Learning. Contributed to the following sections in report writing – Abstract, Section 1, 2, 3.3, 3.6, 4.1.1, 4.1.4, 4.2.1, 4.2.4, 4.3.5, 4.4, 5,6

Preethi (6674957)- Data preparation and data visualization for water quality and mushroom dataset. Implemented Neural Network for all 3 datasets, implemented q-learning algorithm for mountain car environment. Contributed to Abstract, Section 1, 2, 3.4, 3.6, 4.1.5, 4.2.5, 4.3.6, 4.4, 5,6

Shailaja (6679036)- Implemented K-Means and PCA for all 3 datasets. Contributed to implement ILP for Pyrimidine SAR problem and reviewed the report. Contributed to the following sections in the report 1.2, 2.2.1, 2.2.2, 2.2.3, 3.5, 4.1.6, 4.2.6, 4.3.2, 4.3.7, 5, 6.

Swarnika (6683730) – Data preparation and data visualization for water quality and mushroom datasets. Implemented Decision Trees, Gradient Boosting and Random Forest for all 3 datasets. Tested different datasets and created Pyrimidines_B0.pl, B1.pl and Num.CSV files. 10 folds implementation of ILP and different models. Contributed to the following sections of the report: 3.1, 4.1.2, 4.2.2 ,4.3 ,4.3.1 ,4.3.2, 4.3.3 ,5, 6.

References

https://www.doc.ic.ac.uk/~shm/e_coli.html

https://gym.openai.com/envs/MountainCarContinuous-v0/