Text

Description automatically generated

Toronto, Canada

**A Report on**

Module 1 Midweek Project

Predictive Analytics

(ALY 6020)

Guided by:

Prof. Dr. Harpreet Sharma

|  |  |  |
| --- | --- | --- |
| Name of Student | Submitted By:  NUID | Date of submission |
| Parth Shah | 002956963 | 19th January, 2023 |

**Appendix**

1. Importing several Libraries------------------------------------------------------------------------03
2. Loading Dataset-------------------------------------------------------------------------------------04
3. Splitting dataset-------------------------------------------------------------------------------------04
4. Elbow Method---------------------------------------------------------------------------------------05
5. Elbow Method Plot---------------------------------------------------------------------------------06
6. Create a KNN Classifier----------------------------------------------------------------------------06
7. Accuracy of Model----------------------------------------------------------------------------------07
8. Confusion Matrix-----------------------------------------------------------------------------------07

**Introduction**

In the fields of statistics and machine learning, the Iris dataset is well-known. Sepal width, Sepal length, Petal width, and Petal length are the four characteristics that distinguish each of the 150 observations of iris flowers in the dataset. Three iris species—Iris Virginica, Setosa, and versicolor—are used to identify the samples.

**Data Analysis**

Text

Description automatically generated

**Figure 1: Importing several Libraries**

Figure 1 shows importing of several libraries that will be used in the k-NN model including:

* pandas and NumPy for data manipulation and manipulation.
* matplotlib and seaborn for data visualization.
* train\_test\_split and KNeighborsClassifier from sklearn. model\_selection and sklearn.neighbors respectively to split the data into training and test sets and train the k-NN model.
* make\_classification from sklearn.datasets to generate data for classification.
* MinMaxScaler from sklearn.preprocessing to normalize the data.
* accuracy\_score from sklearn.metrics to evaluate the model.
* confusion\_matrix from sklearn.metrics to create a confusion matrix.
* Warnings to filter the warnings.

Text

Description automatically generated

**Figure 2: Loading Dataset**

* This code loads the Iris dataset from scikit-learn's ‘load\_iris()’ function and assigns the feature data to the variable X and the target data to the variable y. This separates the data into two sets, one for the input features (X) and one for the output labels (y).



**Figure 3: Splitting dataset.**

* The model selection module of scikit-learn's model package is used in this code to separate the feature and target data into training and test sets.
* Four inputs are required by the train test split () function: the feature data (X), the target data (Y), the test size (in this example, 0.3, which indicates that 30% of the data will be utilized for testing), and random state (to ensure that the same random samples are selected each time the code is run).
* For training data, it returns two variables, X\_train and y\_train, and for test data, X\_test and y\_test.

Text, letter

Description automatically generated

**Figure 4: Elbow Method**

* This code creates an empty list called "accuracy\_rate". Then, it enters a for loop that iterates from 1 to 29. In each iteration, a KNeighborsClassifier object is created with the number of neighbors set to the current value of the loop variable 'k'. The classifier is then fit to the training data using the 'fit' method.
* Next, it makes predictions on the test data using the 'predict' method, and the accuracy of these predictions is appended to the "accuracy\_rate" list using the 'score' method. This process is repeated for each value of 'k' in the loop, resulting in a list of accuracy rates for each value of 'k' from 1 to 29.

Chart, line chart

Description automatically generated

**Figure 5: Elbow Method Plot**

* In figure 5 visualization, the relationship between the value of K and the accuracy rate, with the goal of identifying the optimal value of K for the K-Nearest Neighbors algorithm on the given dataset. The optimal value of K is the one that results in the highest accuracy rate.



**Figure 6: Create a KNN Classifier**

* This code creates an object of the KNeighborsClassifier class with the number of neighbors set to 10. This means that when the classifier makes a prediction for a new data point, it will consider the 10 closest points in the training data and use their labels to make the prediction.

Text

Description automatically generated with low confidence

**Figure 7: Accuracy of Model**

* This code first fits the KNeighborsClassifier object "knn" to the training data using the "fit" method, with the training data being passed as the first argument and the corresponding labels passed as the second argument.
* Then, it makes predictions on the test data using the "predict" method and assigns the result to the variable "y\_pred".
* Finally, it prints out the accuracy of the model on the test data. The "score" method is used to calculate the accuracy, which returns the mean accuracy on the given test data and labels. The result is multiplied by 100 and formatted to show two decimal places using the string format method.
* Accuracy of the model is 97.78 percent.

Chart

Description automatically generated

**Figure 8: Confusion Matrix**

* Sestosa and Virginica both correctly forecast 100 percent of the time, according to the above confusion matrix, whereas Versicolor predicts correctly 19 out of 18 times and incorrectly 1 out of 19 times.
* k-Nearest Neighbors (k-NN) is a simple, yet robust classification algorithm that can be used on the iris dataset. It is considered a good model for the iris dataset because it is easy to understand and implement, and it often produces highly accurate results.
* The Iris dataset is a small dataset and k-NN is a simple algorithm and it's easy to understand the problem and the solution. For this reason, it is considered a good model for the iris dataset.

**Conclusion**

* In conclusion, k-Nearest Neighbors (k-NN) is a good model for the iris dataset because it is a simple algorithm that is easy to understand and implement, and it often produces highly accurate results.
* The Iris dataset is a small dataset and k-NN is a simple algorithm which makes it a good fit. The accuracy of the model may depend on the choice of the k value, if a good value for k is chosen, it can produce very accurate results.

**Reference**

* sklearn.neighbors.KNeighborsClassifier. (n.d.). Scikit-learn. Retrieved January 19, 2023, from <https://scikit-learn/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
* Elbow Method for optimal value of k in KMeans - GeeksforGeeks. (2019, June 6). GeeksforGeeks. Retrieved January 19, 2023, from <https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/>
* Python Machine Learning - Confusion Matrix. (n.d.). Python Machine Learning - Confusion Matrix. Retrieved January 19, 2023, from <https://www.w3schools.com/python/python_ml_confusion_matrix.asp#:~:text=What%20is%20a%20confusion%20matrix,the%20predictions%20we%20have%20made>.