Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

536 : APPLIED MACHINE LEARNING

PROJECT 1 : EXPLORING LABELED DATA USING KNN

PROJECT REPORT

1. Problem Description: The objective of this project is to understand the workflow of supervised learning by utilizing KNN classifier to explore a labeled dataset (Penguins dataset is used in this project). Implement K-nearest neighbors machine learning algorithm and evaluate the model's performance using confusion matrix, accuracy, classification report etc. In this project, penguins' species is predicted based on the anatomical information using K-nearest neighbors machine learning algorithm and evaluated the accuracy of the classification model.

2. Background of dataset: The Penguins dataset is imported from the seaborn library. It consists of seven columns namely species, island, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g and sex. Among these, species is a target variable (dependent variable) and remaining are independent variables. Below figures show the sample data and class distribution.

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female

```
penguins['species'].value_counts()
```

```
Adelie       146
Gentoo       119
Chinstrap     68
Name: species, dtype: int64
```

Supervised learning algorithms try to predict a target (dependent variable) using features (independent variables). Depending on the characteristics of target variable, it can be a classification (discrete target variable) or a regression (continuous target variable) task.

Data Wrangling: Missing values present in the dataset are negligible in count and are handled by deleting the rows with missing values.

3. Statistical Summary:

Statistical summary of attributes with respect to each class:

Adelie

	Bill Length(MM)	Bill Depth (MM)	Flipper Length(MM)	Body Mass (G)
Count	146	146	146	146
Mean	38.824	18.347	190.00	3706.00
STD	2.663	1.219	6.522	458.620
Min	32.1	15.500	172.00	2850.00
Max	46.00	21.500	210.00	4775.00
Median	38.85	18.400	190.00	3700.00
Mode	51.3	17.3	195.00	3800

Chinstrap

	Bill Length(MM)	Bill Depth (MM)	Flipper Length(MM)	Body Mass (G)
Count	68	68	68	68
Mean	48.833	18.42	195.82	3733.088
STD	3.339	1.135	7.131	384.335
Min	40.90	16.40	178.00	2700.00

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

Max	58.00	20.80	212.00	4800.00
Median	49.55	18.45	196.00	3700.00
Mode	51.3	17.3	195	3650

Gentoo

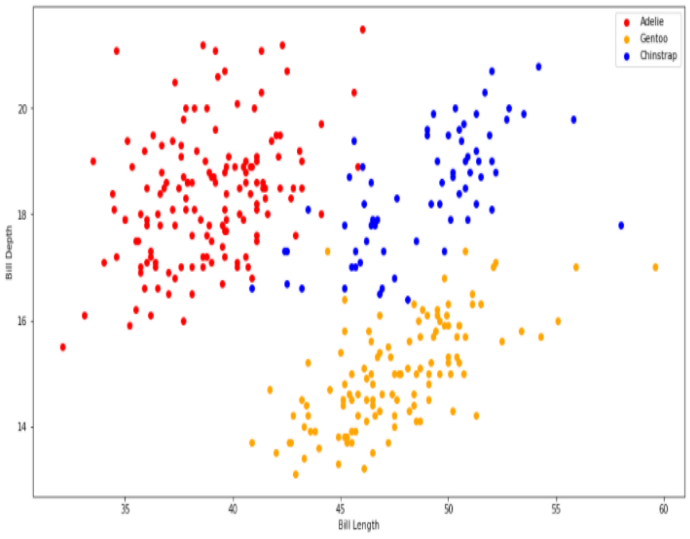
	Bill Length(MM)	Bill Depth (MM)	Flipper Length(MM)	Body Mass (G)
Count	119	119	119	119
Mean	47.568	14.996	217.235	5092.436
STD	3.106	.986	6.585	501.476
Min	40.90	13.10	203.00	3950.00
Max	59.60	17.30	231.00	6300.00
Median	47.40	15.00	216.00	5050.00
Mode	50	15	215	5550

Statistical summary of categorical data:

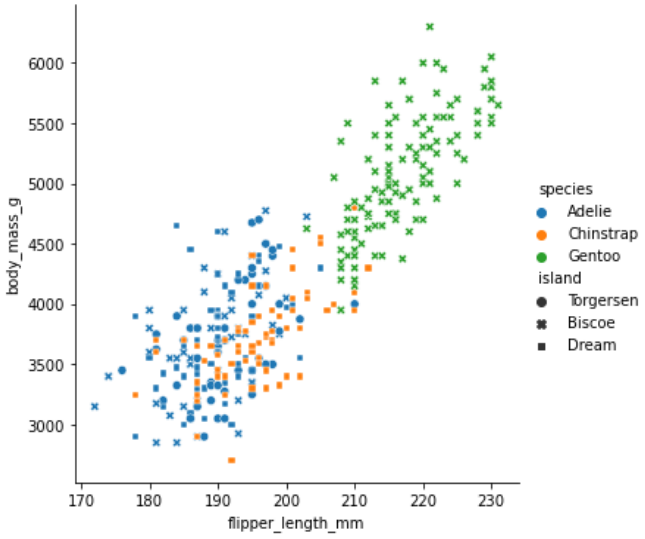
		species	island	sex
species				
Adelie	count	146	146	146
	unique	1	3	2
	top	Adelie	Dream	Male
	freq	146	55	73
Chinstrap	count	68	68	68
	unique	1	1	2
	top	Chinstrap	Dream	Male
	freq	68	68	34
Gentoo	count	119	119	119
	unique	1	1	2
	top	Gentoo	Biscoe	Male
	freq	119	119	61

Data Visualization:

Bill Length vs Bill Depth vs Species



Body mass Vs FlipperLength Vs Species Vs Island



4. K-nearest Neighbor Algorithm:

K-nearest neighbors is an algorithm which classifies a new data point based on its proximity to other data point groups. Higher the proximity of new data point from one group, higher is the likelihood of it getting classified into that group. Distance between data points is measured by distance metrics like euclidean distance, manhattan distance, minkowski distance etc. It is a supervised learning machine learning algorithm that deals with labeled data which means the

data is tagged with labels or tags. The machine is provided with a set of data(training) so that supervised learning algorithm analyses the training dataset and produces a correct outcome from labeled data. Thus, the machine learns the things from training data and then applies the knowledge to test data.

Unsupervised learning algorithm tries to find the structure of unlabeled data i.e. group unsorted information according to similarities, patterns, and differences without any prior training of data.

Application of KNN algorithm:

- Split dataset into training and testing. Divided the penguins dataset with a split ratio of 80-20 that is 80% training data and 20% testing data.
- **Feature Scaling:** Performance of KNN algorithm can be improved by feature scaling. In KNN, all numerical features should have the same scale. If you pay attention, the Body Mass Index column is at a very different scale than all other body measurements. It will be unfair to calculate distance between such type of values. So, all features should be scaled so that they range from 0 to 1. This can be done using a method called StandardScaler (called normalization)
- Decide the k value.
- If we use weights as uniform then the new data point is categorized based on maximum number of k similar points surrounded to it. If we consider weights as distance then the new point is classified as the shortest distance between new data point and k nearest neighbors.
- Calculate the accuracy of the test dataset and check for stability in accuracy. If the accuracy is not acceptable, then this algorithm may not be best for this dataset.

  To decide the best k value, weights, and other parameters, different combinations were tested on the testing dataset and got a best accuracy of 98.5%. The different parameters used are as below.

```
n_neighbors = [5,7,9,11,13,15,17,19,20]
weights = ['uniform', 'distance']
algorithms = ['auto', 'brute', 'ball_tree', 'kd_tree']
leaf_sizes = [10,20,30,40]
p_values = [1, 2]
metrics = ['euclidean', 'manhattan', 'chebyshev', 'minkowski']
```

```
for number in n_neighbors:
    for weight in weights:
        for algorithm in algorithms:
            for size in leaf_sizes:
                for p_value in p_values:
                    for metric in metrics:
                        currentKNN = KNeighborsClassifier(n_neighbors=number, weights = weight, algorithm=algorithm, leaf_size=si
                        currentKNN.fit(X_train, y_train)
                        currentScore = currentKNN.score(X_test, y_test)
                        if currentScore > bestScore:
                            bestScore = currentScore
                            bestKNN.clear()
                            bestKNN.append(currentKNN)
                            #print(bestScore)
                            print("Best KNN Parameters are : k =",number,"metric =",metric, "Weights =", weight, "Algorithm =", a

print("Best Accuracy of the model is: ", bestScore*100)
```

The best combination for our dataset is {n_neighbors=5, weights = 'distance', algorithm='auto' , leaf_size=10, p=1, metric='manhattan'} with an accuracy of 98.5 %.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

5. Summary of Classification results:

Accuracy of the model : 98.5%

Classification Report: Classification report is used to measure the quality of predictions from a classification algorithm like how many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

	precision	recall	f1-score	support
Adelie	0.97	1.00	0.99	36
Chinstrap	1.00	0.92	0.96	13
Gentoo	1.00	1.00	1.00	18

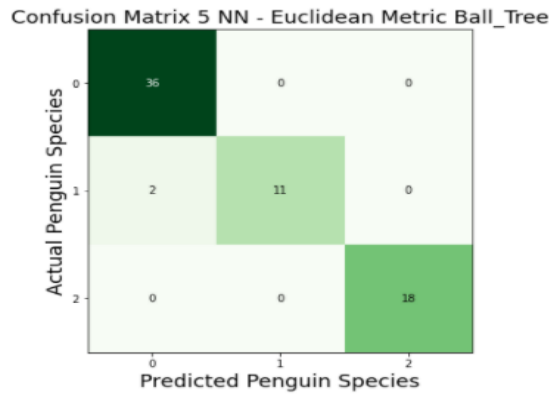
Confusion matrix: This allows you to visualize the performance of the classification machine learning models and is better metric than accuracy to validate a model. Below is the confusion matrix of our dataset where horizontal values indicate predicted species and vertical values indicate actual species.

```
[[36  0  0]
 [ 1 12  0]
 [ 0  0 18]]
```

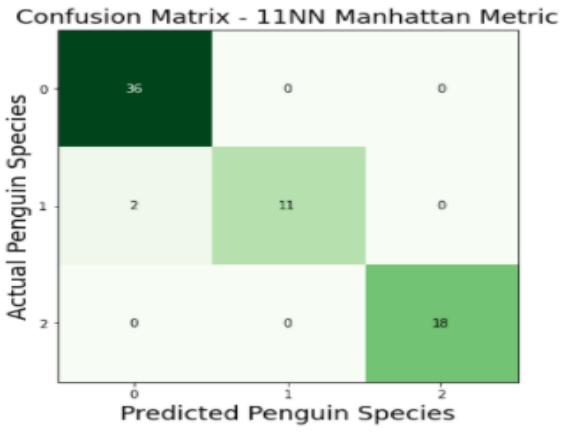
The true value of Chinstrap species is predicted as Adelie species. Similarly true value Gentoo species is correctly predicted as Gentoo species and so on.

Confusion Matrices for few distance metrics and k values:

```
Confusion Matrix 5NN - Euclidean Metric - Ball_Tree
[[36  0  0]
 [ 2 11  0]
 [ 0  0 18]]
Accuracy:  97.01492537313433
```



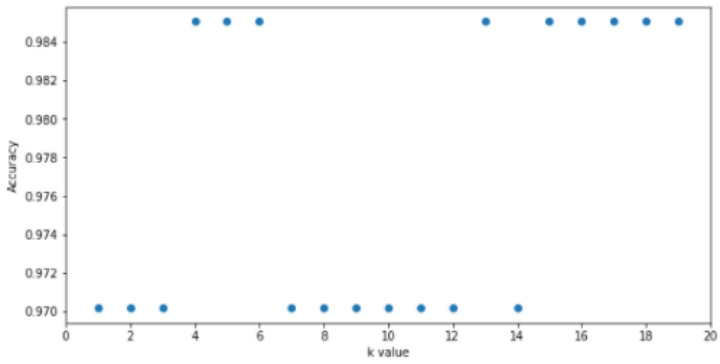

```
[[36  0  0]
 [ 2 11  0]
 [ 0  0 18]]
Accuracy:  97.01492537313433
```



6. Discussion of results:

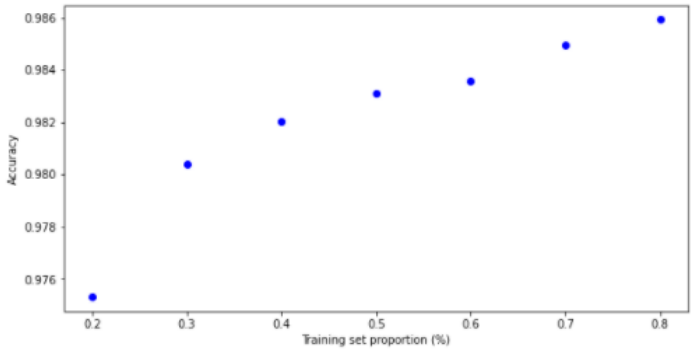
- **Distance Metric:** The distance metric that we believe works best was Manhattan, but the difference was very small when compared to other metrics. With other metrics we got around 97% accuracy rate and with manhattan metric we achieved 98.5% accuracy.
- **Accuracy Vs K value:** At k=4,5,6 knn model achieved best accuracy of 98.5% and then again at k=13,15 and so on accuracy is maximum i.e. 98.5%. At remaining k values, accuracy is around 97%.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263



- **Accuracy Vs Dataset Partition:** From the below plot, it can be observed that accuracy increases with increase in training dataset size.

  Having a large training subset help us in getting better accuracy as machine is trained more. On the other hand, it becomes computationally expensive, as the algorithm stores all the training data to test a new data point.



- **Data leakage problem:**

  After dataset partition into training and testing, if some of the data from testing is shared to training dataset and vice versa, then whatever model we are going to use will give us high accuracy with respect to test data. But when deployed in production, the model will not give best accuracy because of data leakage.

  Data Leakage happens in two scenarios:

  a) If there are any missing values in the dataset then we usually fill those missing values with mean or median values of the entire dataset (which involves test data also). To avoid this problem, we should first split dataset into training and testing and then replace NaN values with their respective mean or median values.

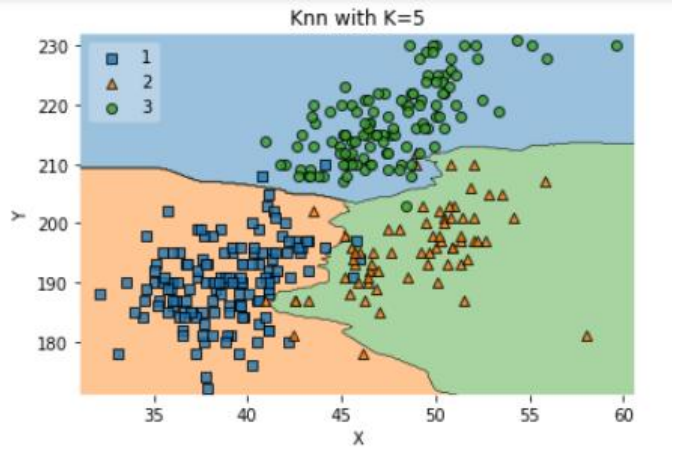
  b) It occurs in time series data problems. In times series there is some dependency between attributes. If we randomly split dataset as train and test then data leakage problem may occur. To avoid this, we find a point p and group all the values before p as training and after p point as testing data.

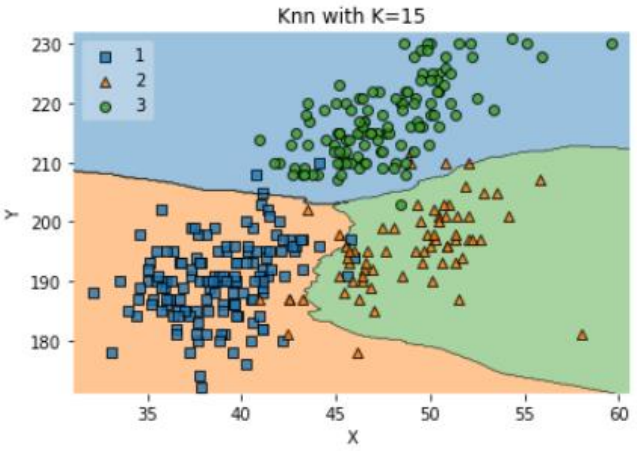
  In our penguins dataset, we do not have missing values and no time series data. Hence there is no data leakage problem.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

7. Conclusion of our exploration:

There is a high success rate in our penguins dataset when classifying different species of penguins based on the anatomical information. It can be said that KNN classification algorithm is suitable for our dataset. Below are some sample plots to visualize KNN for 3-class classification. Increase in border smoothness can be observed with increase in k value.





Pros and Cons of KNN:

- KNN is simplest and lazy learning algorithm because it requires no training prior to making real time predictions, instead it stores complete dataset. All the computation happens during scoring i.e. when we apply the model to predict. This makes KNN faster than other algorithms that require training like SVM, linear regression etc.
- Only two important parameters required to implement KNN are n_neighbours (k value) and distance metric.
- KNN doesn't work well with high dimensional data because it becomes difficult for the algorithm to calculate distance in each dimension and the cost of calculating distance between new point and each existing point becomes higher.
- KNN doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

8. Summary of commonly used nearest neighbor finding algorithms:

The principle behind nearest neighbor algorithms is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning).

Scikit-learn in Python implements two different nearest neighbors classifiers: KNeighborsClassifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. RadiusNeighborsClassifier implements learning based on the number of neighbors within a fixed radius r of each training point, where r is a floating-point value specified by the user.

Sai Jyothi Ongole
so70@uakron.edu
Student ID : 4785263

Nearest Neighbor Algorithms:

** Here N and D indicate samples and dimensions respectively.

a) **Brute:** This algorithm performs the distance calculation of all the N training points in D dimension and the Big O notation is given as O[DN^2]. This kind of search is efficient for small size data sets. When the size of data points increases this algorithm becomes inefficient.

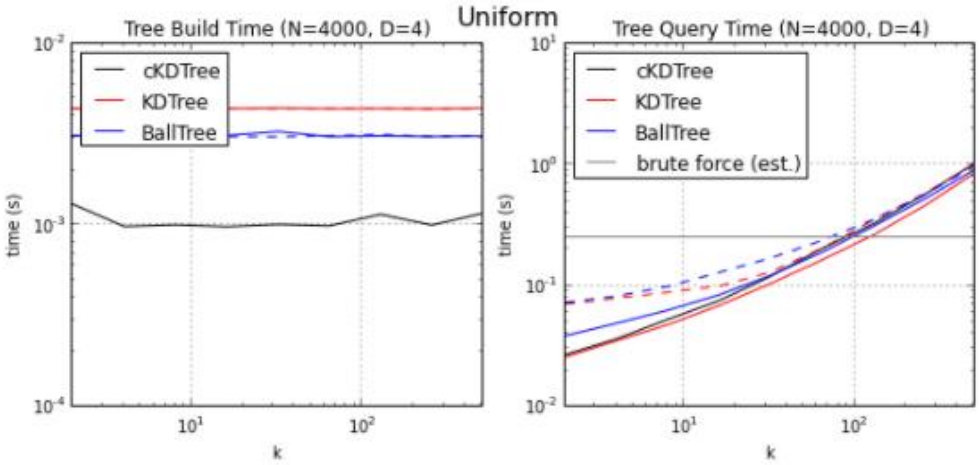
b) **K-D Tree:** This algorithm is used to address the computational inefficiencies of Brute force. In this algorithm, the distance between two points is measured, and if the third point is far away from second point then it is obvious that third point is also far from first point. Hence the distance between first and third points are not measured. In this way, the computational cost can be reduced to O [DN log(N)] or better. This approach is fast for low dimensional D<20 neighbors search, as D grows, this algorithm becomes inefficient.

c) **Ball Tree:** To address the inefficiencies of K-D Tree, ball tree is developed, where ball tree divides data in a nesting hyper sphere. This makes it very efficient, even in high dimensions. It divides the data recursively into nodes defined by centroid C and radius r, such that each node lies in the hyper sphere defines by r and C.

d) **Auto:** It is the default value; the algorithm determines the best approach from training data.

e) **CKD Tree:** cKDTree uses explicit dynamic memory allocation at the construction phase. This means that the trained tree object cannot be pickled, and must be re-constructed in place of being serialized. Because of the flexibility gained through the use of dynamic node allocation, cKDTree can implement a more sophisticated building methods: it uses the "sliding midpoint rule" to ensure that nodes do not become too long and thin.

Below plot shows the comparison of performances of three trees Ball tree, KD tree and cKD tree.



Referenced Websites:

- https://www.towardsdatascience.com
- https://scikit-learn.org
- https://jakevdp.github.io/blog/2013/04/29/benchmarking-nearest-neighbor-searches-in-python/