Course Code: CT4101 (Machine Learning)	Assignment: 1
Name: *Seshadri Sundarrajan*	Student ID: 19230107

CLASSIFICATION REPORT (DATASET: HAZZLENUTS)

MACHINE LEARNING PACKAGE USED:

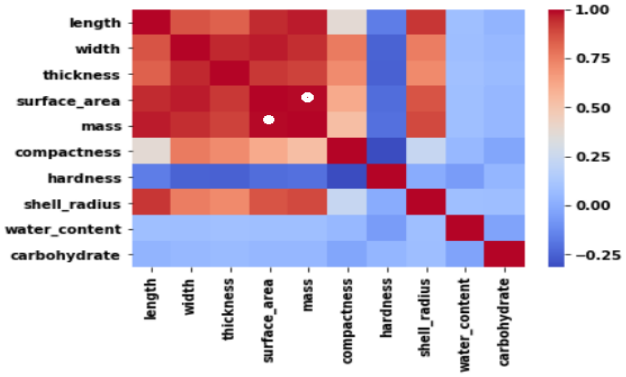
- Scikit-learn (Open source tool for data mining and analysis in python)
  - Wide usage across the industry because of its API documentation and community support.
  - All in one package, as it provides sub modules for both supervised and unsupervised algorithms along with model selection, dimensionality reduction and pre-processing. This feature comes in handy as it caters to most of our needs by eliminating inter-library dependencies that we face while dealing with different packages for a eachneed.
  - Pipelining capability is more useful while designing and altering end to end algorithms flow.
  - Apart from the above reasons, the most important factor choosing this package is because of its interface with Python, which is more comfortable due to its readability, robustness and accessibility.
- Matplotlib and Seaborn are used for visualization while Pandas and Numpy are used for manipulation. (IDE – Jupyter Notebook)

DATA PRE-PROCESSING :

- The provided dataset had columns representing instances. To feed into Pandas data frame and for ease of interpreting *transposed* the data to get instances along the row.
- Performed Min Max scaling on the training set to get each feature values between 0 and 1. When dealing with distance metrics if scaling is not done then features on higher scale dominates the classification. Train and Test data are split under ratio 3:1.

FEATURE SELECTION:

- 'Sample id' column has no impact on our classifier as it's just a key, so dropped it.
- Performed label encoding on the 'variety' column to get numeric class values.
- Features 'Surface area' and 'Mass' had the highest correlation of 0.994, therefore removed '*Surface area'* to eliminate multicollinearity and redundancy.
- This heat map visualizes correlation matrix, where above-mentioned features are highlighted.



MODEL - 1 : KNN

- One of the simplest supervised machine learning algorithms with less calculation time and better predictive power.
- The Algorithm operates by mapping the data points in the vector space as part of training. Thus, training time is negligible.
- KNN is also non-parametric (do not make any assumption for mapping function, free to learn this function from data) and lazy algorithm (computation occurs at testing time)

- Later while predicting classes for test data, algorithm maps the data point on the trained data space and finds the 'k' nearest points to our test data.
- K nearest points are found by
    - Calculating the distance between test data and each instance of training data.
    - Sorting the calculated distances in ascending order based on distance values.
    - Picking top k rows from the sorted array
- Final evaluation could be mapping test data to a class that has majority of votes from k nearest points.

PARAMETERS TWEAKED:

- **K-value**: It can be decided only by analyzing performance for different k - values to get ideal value.
- **Weights**: Prediction method can be controlled by assigning uniform or distance - based weights to neighbours. We can assign higher weights to the closest neighbour.
- **Distance Metric**: Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Manhatton Chebyshev, cosine, etc.

Values: [Distance metric = Euclidean, K Value = 7, Weights = Distance based]

METRICS:

- Constructed model based on above specification yields an accuracy of 94%.
- Confusion matrix and classification report are shown below from which we can infer that for the class '*Avellana' and 'Cortuna'* our model prediction is 100%.

Confusion Matrix

	c_avellana	c_americana	c_cornuta
c_avellana	14	2	0
c_americana	0	17	0
c_cornuta	0	1	17

Classification Report

	precision	recall	f1-score	support
c_avellana	1.00	0.88	0.93	16
c_americana	0.85	1.00	0.92	17
c_cornuta	1.00	0.94	0.97	18
accuracy			0.94	51
macro avg	0.95	0.94	0.94	51
weighted avg	0.95	0.94	0.94	51

REFERENCES:

1. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
2. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

MODEL - 2 : XGBOOST

- XgBoost is one of the most dominating machine learning algorithms recently. It suits for both regression and classification.
- It is specifically known for its hardware utilization and prediction power by its parallel and distributed computing along with cache optimization.
- It also supports regularization parameters to penalize models as they become more complex and reduce them to simple models.
- Extreme Gradient Boosting is an ensemble model that makes use of several weak learners (decision trees) to get better prediction sequentially

- Training phase involves
    - Fitting training data upon our first weak learner and then data points that are wrongly classified are given higher weights.
    - Later these highly weighted data points are sent to fit in second weak learner along with other points. Now again higher weights are assigned for misclassified data.
    - The above two process takes place sequentially until we arrive at our final classifier.
- When the model is tested, all our weak learners classify test data and a final decision is class that is voted majorly.

PARAMETERS TWEAKED:

- **Max depth:** We can control the maximum Tree depth of our weak learners.
- **N estimators:** Number of weak learners to fit.
- **Learning rate:** Shrinkage amount for each descending operation to arrive at an optimal value. Lesser the value greater the accuracy but increases computational steps to reach global minima.
- **Subsample:** We can specify the percentage of samples used per tree.

Values: [Max depth = 5, N estimators = 75, Learning rate= 0.01, subsample = 0.7]

METRICS

- Model overall accuracy is 90%.
- Confusion matrix and classification report are shown below from which we can infer that for the classes '*Avellana' and 'Cornuta'* our model prediction is 100% but for the class '*Americana*' it's only 77%.

Confusion Matrix

	c_avellana	c_americana	c_cornuta
c_avellana	11	5	0
c_americana	0	17	0
c_cornuta	0	0	18

Classification Report

	precision	recall	f1-score	support
c_avellana	1.00	0.69	0.81	16
c_americana	0.77	1.00	0.87	17
c_cornuta	1.00	1.00	1.00	18
accuracy			0.90	51
macro avg	0.92	0.90	0.90	51
weighted avg	0.92	0.90	0.90	51

REFERENCES

1. https://www.datacamp.com/community/tutorials/xgboost-in-python
2. https://xgboost.readthedocs.io

K -FOLD CROSS VALIDATION

- While fitting our training data to the model we might be splitting our data in such a way that we might omit certain data that exhibit new behaviour and get split into test data.
- When the same model is judged with test data it performs poorly as completely new behaviours get exhibited which haven't recorded while training. This can be overcome using k -fold cross-validation.

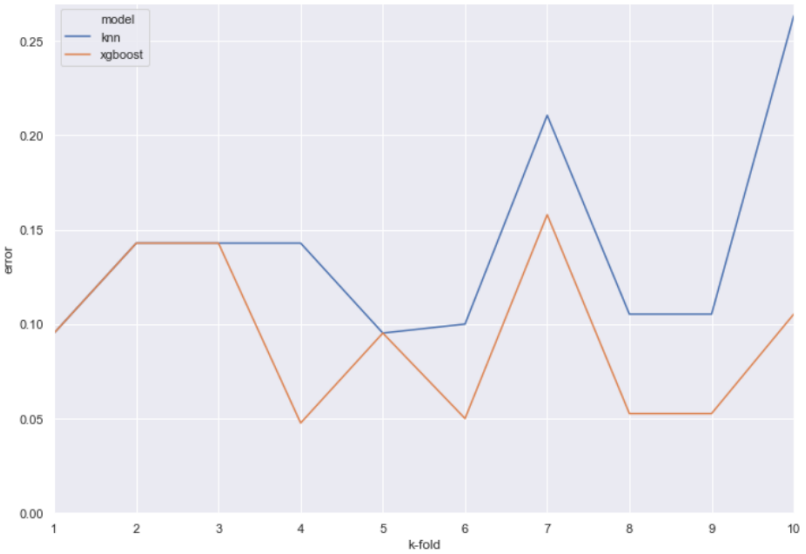
METHODOLOGY

- K -fold cross validation helps in splitting our dataset into k folds(partitions).
- At each iteration one partition is assigned for testing and rest are used for training.
- We fit the model and predict our accuracy from the test fold.
- Across all the accuracies obtained from all iterations, we can find the best fold that splits our data perfectly.

MODEL EVALUATION WITH K-FOLD

Plotted the misclassification error for both the models using 10 – fold cross validation.

- Both the models had overlapping behaviour from 1st to 3rd fold.
- KNN got lowest error as 9% at 1st and 4th fold and got mean accuracy of 85%.
- XGBoost had lowest error as 5% at 4th,6th,8th and 9th fold with mean accuracy of 95%.



INFERENCE

1. Initially, KNN outperformed XGBoost by 4 % accuracy but after performing cross validation and finding out mean accuracy, XGBoost had better accuracy that is nearly 10% greater than KNN.
2. XGBoost has better performance than KNN in our scenario due to its underlying concept of weak learner sequencing. Here, KNN is a weak learner and XGBoost pools in several weak learners to arrive final model with high efficiency. Also, in real-time with large amount of data, XGBoost is reliable and time saving because of its computing capabilities (Where KNN performs badly as it requires high memory and it is computationally expensive)
3. Decision surface of both models are displayed below (Used only two features for ease of visualization, where XGBoost has more linear boundary)