Fruit classification using ML techniques

*

1st Ashwani
Computer Science and Artificial Intelligence
Indraprastha Institute of Information Technology
Delhi, India
ashwani21521@iiitd.ac.in

2nd Devesh Kumar Shaw
Computer Science and Artificial Intelligence
Indraprastha Institute of Information Technology
Delhi, India
devesh21526@iiitd.ac.in

***Abstract*—In this project, we use many machine learning techniques to predict the category of test data on the basis of given train data. Data is given in two csv files as "train.csv" and "test.csv". Train data contains 4098 columns with 1st and last column as 'ID' and 'category' respectively. Test data contains 4097 columns with 1st column as 'ID' and we are to predict the 'category' column for this. Column 'category' consisting of 20 unique values in strings. We perform label enconding, Isolation forest, K-Means clustering, PCA, RandomForest classifier and Logistic Regression to predict the category for test data.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In this report, we discuss about the methods and techniques we used to predict and generate the predicted categories for the test dataset.

II. Methodology

A. Data Loading

To begin the data analysis process, the provided CSV files were transformed into pandas dataframes. The "train.csv" file consists of 4098 columns, with the first and last columns representing the "ID" and "category" respectively. The "category" column is comprised of 20 distinct strings. In contrast, the "test.csv" file contains 4097 columns since it lacks the "category" column. Furthermore, the "train.csv" file was divided into training labels and the remaining data for further analysis. This step was taken to facilitate the creation of models that can accurately predict the category of data in the "test.csv" file.

B. Label Encoding

Since, category comprised of strings, we have to convert them to numeric data to move further. Hence, we use LabelEncoder() to convert data in category to numeric form so we can perform further techniques on it.

C. Outlier Detection

In our approach, we used the "IsolationForest()" function from the "sklearn.ensemble" module in python to detect and remove any outliers from the dataset. Isolation Forest works by identifying observations that are far away from any clusters in the data and isolating them as outliers. By removing these outliers, we can focus on the more relevant data points, which can improve the accuracy of our model.

Identify applicable funding agency here. If none, delete this.

D. Clustering

To improve the classification of data points, we used K-Means clustering to group the training data into 20 distinct clusters. The category column in the dataset consisted of 20 unique values, so we opted to create the same number of clusters. By doing this, we could add additional features to the test data that correspond to the cluster to which each data point belongs.

E. Dimension Reduction

Other than 'ID' and 'category' we still have 4094 columns of data therefore it is required to decrease the number of features of data to increase the complexity of our model. We used PCA (Principle component analysis) to reduce the dimension of data. The number of new features is set to 100.

F. Ensemble

Ensemble learning is a technique used in machine learning to combine multiple models to improve overall performance. In our project, we used the "VotingClassifier" method from the "sklearn.ensemble" module to ensemble our models. We combined two classifiers - 'LogisticRegression()' and 'RandomForestClassifier()' - to increase the accuracy of our predictions. We implemented the soft voting method in our ensemble, which means that we used the averaged predicted probabilities of each classifier to make the final prediction. Soft voting is an effective method to combine the output of multiple models as it takes into account the confidence of each model's prediction. By combining these models, we were able to achieve better accuracy in our predictions.

G. Kfold

We used k-fold cross-validation to check the performance of the model. The k-fold cross-validation idea divides the data into k equal parts, then trains k-1 folds of data and tests the remaining data. This process repeats k times, and every fold gets to try once. This algorithm is used to check which model is better and find the accuracy of the models. This algorithm is also used to tune the hyperparameters of a model. K-fold is also used for outlier detection as we can run the model with and without outliers to find out. We used sklearn.modelselection library to import the Kflod function, and we split data into five folds and passed it into as a parameter in the function and created an array that will store the scores of prediction and folds. Later we take the mean of the array, and it will give the possible score of the model. The Cross-validation accuracy of our model is 77.22

Acknowledgment

We want to express our sincere thanks to Professor Koteswar Rao Jerripothula for his invaluable guidance and support. His help and feedback helped us to improve our work and guided us to the success of this project. And we are grateful to our TA's and TFs for guiding us. And finally, we would like to acknowledge Kaggle for providing a stage for this project.

References

[1] scikit-learn.RandomForest Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[2] scikit-learn. LabelEncoder. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[3] scikit-learn. IsolationForest. Retrived from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

[4] scikit-learn. KMeans. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[5] scikit-learn. PCA. Retrieved from https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html