PESU-RR-TEAM52-106-130-918

Dataset Analysis Questions

Q1: Selecting a Validation Split Method

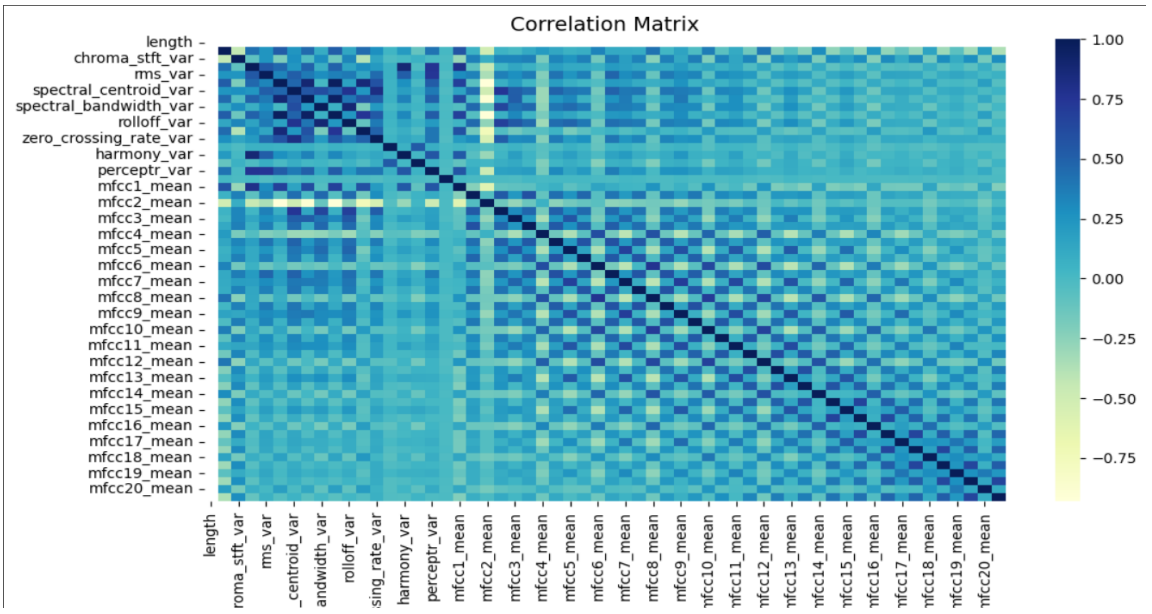
• What is a good validation split method, and what are you planning to use for this dataset, and why?

Ans. A good validation split method for our dataset would be the 80-20 split for training and validation. We use this method to check how well our model's Learning function has been trained and whether it can work well with unseen data. It also helps with hyperparameter tuning, as we can fine tune our model based on the validation accuracy to attain maximum model accuracy. We use this method for the given dataset as we have to predict the values using test.csv file, thus using this method helps in understanding how well the model generalises to new, unseen examples. We used the train_test_split function from scikit-learn library to split the dataset.

Q2: Exploring Column Correlations and Impact on Accuracy

• Is there a correlation between any of the columns? If so, does removing one of the columns affect the accuracy, and how?

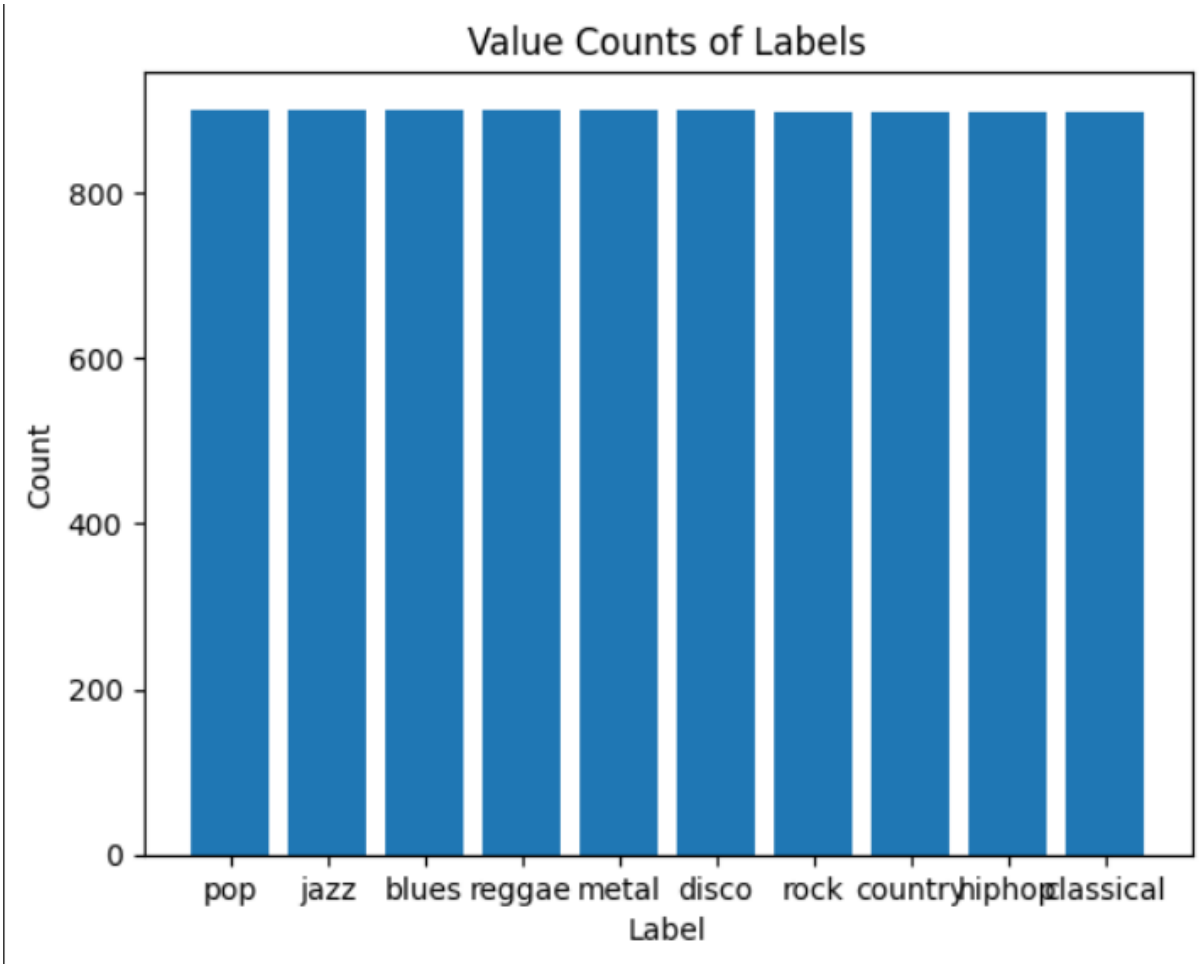
There is a correlation between the columns of the dataset as indicated in the heatmap below-



Removing some columns in the dataset doesn't affect accuracy, like 'filename', 'length', as it doesn't contribute to the output label. Removing columns that are weakly correlated, increases accuracy, as indicated in our notebook. We reduced the number of input features from 58 to 30, initially based on correlation scores and importance scores from the RandomTree Classifier for feature importance, the accuracy increased significantly. In our final model we used 35 input features, which gave the maximum accuracy score, which is indicated on our Kaggle score.

Q3: Analyzing Class Distribution and Addressing Imbalance

- Plot the distribution of the Genres and discuss if there is any class imbalance.
- In case there is an imbalance, how would you solve it? List two methods.



As indicated in the plot above, we can see that the labels are not imbalanced in the train.csv file. Thus there is no class imbalance in the dataset.

```
pop          900
jazz         900
blues        900
reggae       900
metal        900
disco        899
rock         898
country      898
hiphop       898
classical    898
Name: label, dtype: int64
```

In case there was an imbalance, we can use the following methods:-

1. Oversampling: In oversampling, you generate more instances of the minority class to balance the dataset. This can be done by duplicating existing samples or by generating synthetic samples using techniques like SMOTE (Synthetic Minority Over-sampling Technique).
2. Undersampling: Undersampling involves reducing the number of instances in the majority class to match the minority class. This can help balance the dataset but may result in a loss of information.
3. Using Weighted Loss or Cost-sensitive Learning: Assign different weights to classes during training. Most machine learning algorithms allow us to assign weights to classes based on their prevalence. By assigning higher weights to minority classes, we can make the model pay more attention to those classes during training.

Q4: What is overfitting and how will you address it?

Overfitting is a problem in machine learning where a model learns the training data too well, capturing noise and random fluctuations in the data rather than the underlying patterns. As a result, the overfitted model performs exceptionally well on the training data but poorly on unseen or validation data. We have to do the following to address overfitting:-

1. Feature Selection: Carefully select and engineer the most relevant features while removing irrelevant or noisy ones. Feature selection can help reduce the dimensionality of the data and prevent overfitting.
2. Early Stopping: Implement early stopping during model training. Monitor the model's performance on a validation set, and if the performance

starts to degrade (indicating overfitting), stop training to prevent the model from fitting noise.
3. Hyperparameter Tuning: Experiment with different hyperparameter settings, such as learning rates, batch sizes, and the strength of regularization, to find a balance that minimizes overfitting.

Q5: What is underfitting and how will you address it?

Underfitting is a problem in machine learning, and it occurs when a model is too simple to capture the underlying patterns in the data. An underfit model performs poorly on both the training data and unseen or validation data because it fails to learn the relevant relationships.
We have to do the following to address underfitting:-

1. Increase Model Complexity: we should use a more complex model with a larger number of parameters. For example, in the case of neural networks, we can increase the number of layers or units in each layer.
2. Feature Selection: Carefully select and engineer the most relevant features while removing irrelevant or noisy ones. Feature selection can help reduce the dimensionality of the data and prevent underfitting.
3. Hyperparameter Tuning: Experiment with different hyperparameter settings, such as learning rates, batch sizes, and the strength of regularization, to find a balance that minimizes underfitting.
4. Ensemble Methods: We can use ensemble methods like bagging (e.g., Random Forests) or boosting (e.g., AdaBoost) to combine multiple models. Ensemble methods can improve model performance by aggregating predictions from multiple models, which can help address underfitting.