MI Hackathon

Team – Team-194

Member 1 – PES1UG21CS474

Member 2 – PES1UG21CS704

Member 3 – PES1UG21CS501

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?

A simple train-test split which randomly splits the dataset was used for splitting the training data to training and validation datasets. The train-test split was 80%-20% of the whole training data.

A simple train-test split was used because the size of the dataset seemed large enough with 8991 samples and around 54 attributes and assuming equal class distribution.

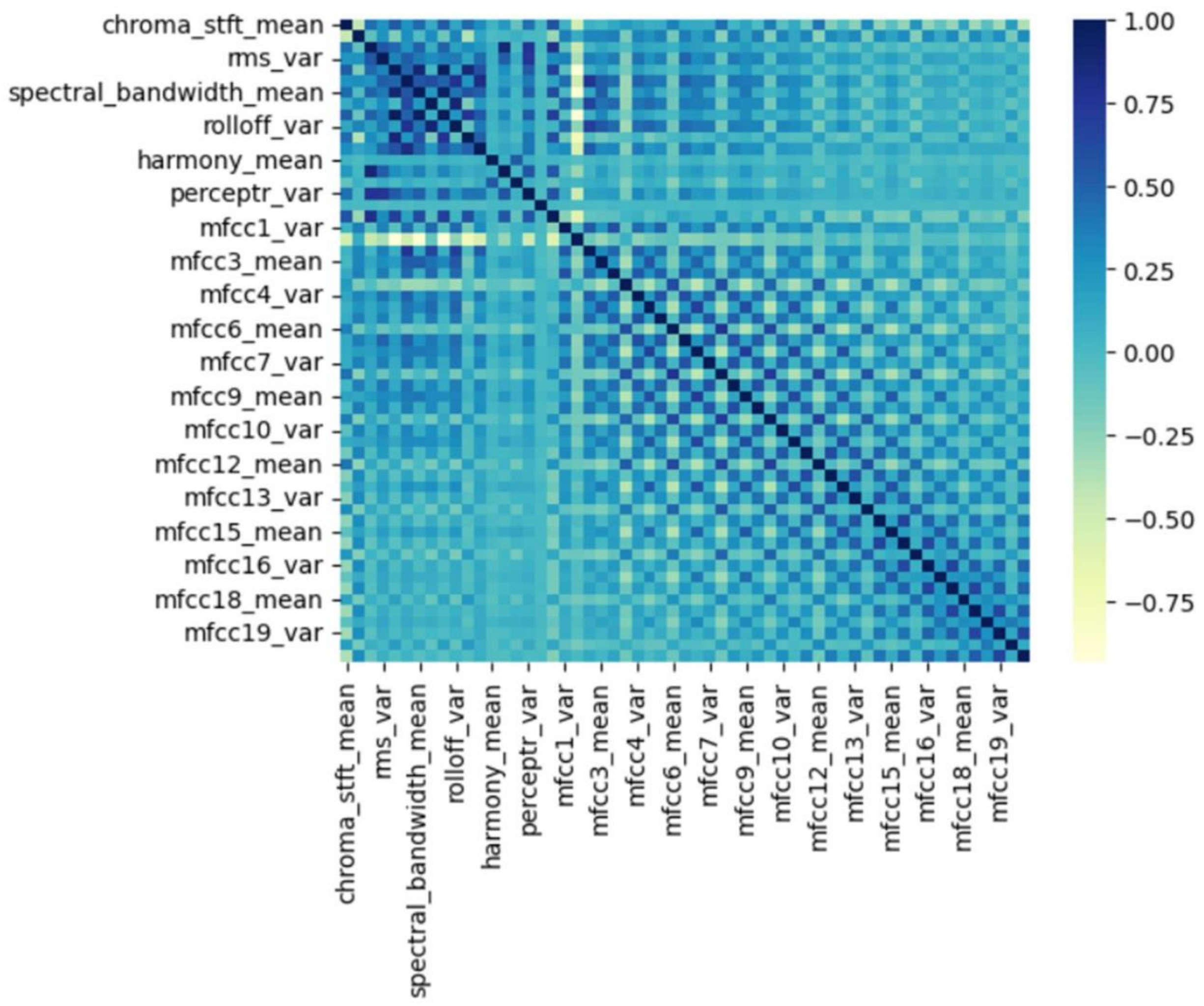
If the training data size were to be smaller k-fold cross-validation could be used to split the data into k subsets and train the model k times.

If the class distribution was not uniform, as it might occur in smaller datasets, stratified sampling would have been a better choice.

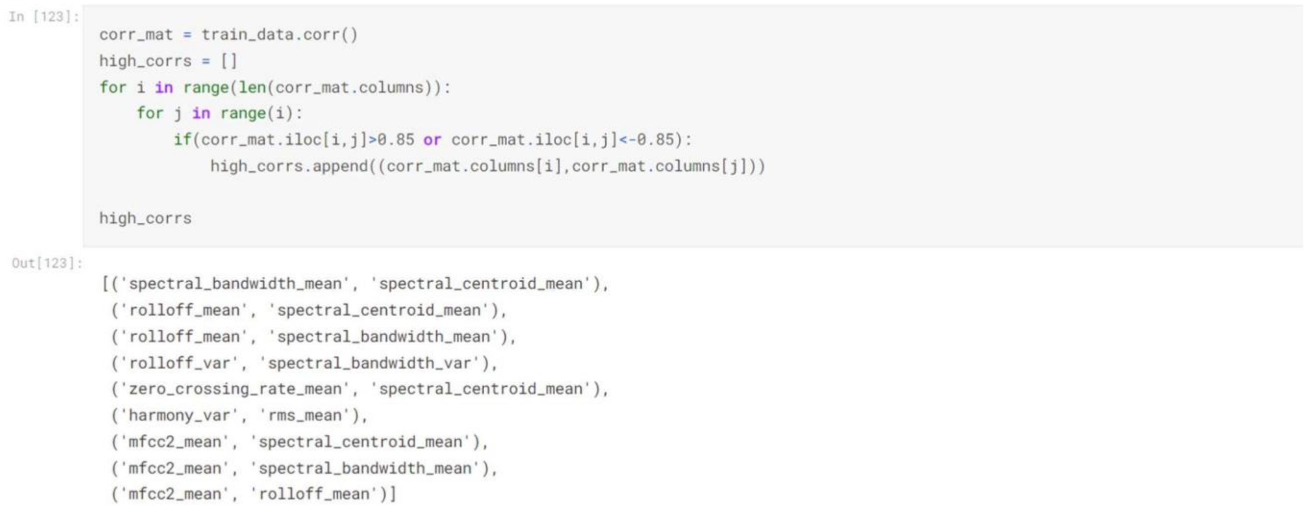
Q2: Exploring Column Correlations and Impact on Accuracy

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

Since there were around 60 attributes for each sample in the dataset, a correlation matrix was derived to remove columns which were correlated.



On analysing correlation between columns individually and filtering out columns with correlation greater than 0.85, these were the results.



```
In [123]:
corr_mat = train_data.corr()
high_corrs = []
for i in range(len(corr_mat.columns)):
    for j in range(i):
        if(corr_mat.iloc[i,j]>0.85 or corr_mat.iloc[i,j]<-0.85):
            high_corrs.append((corr_mat.columns[i],corr_mat.columns[j]))

high_corrs
```

```
Out[123]:
[('spectral_bandwidth_mean', 'spectral_centroid_mean'),
 ('rolloff_mean', 'spectral_centroid_mean'),
 ('rolloff_mean', 'spectral_bandwidth_mean'),
 ('rolloff_var', 'spectral_bandwidth_var'),
 ('zero_crossing_rate_mean', 'spectral_centroid_mean'),
 ('harmony_var', 'rms_mean'),
 ('mfcc2_mean', 'spectral_centroid_mean'),
 ('mfcc2_mean', 'spectral_bandwidth_mean'),
 ('mfcc2_mean', 'rolloff_mean')]
```

The model after removing these columns gave a smaller accuracy of around 68%-72%

Later the columns with correlation greater than 0.9 were filtered out which a gave an accuracy of about 76% with an ANN.

```python
corr_mat = train_data.corr()
high_corrs = []
for i in range(len(corr_mat.columns)):
    for j in range(i):
        if(corr_mat.iloc[i,j]>0.90 or corr_mat.iloc[i,j]<-0.90):
            high_corrs.append((corr_mat.columns[i],corr_mat.columns[j]))

high_corrs
```

```
[280… [('rolloff_mean', 'spectral_centroid_mean'),
       ('rolloff_mean', 'spectral_bandwidth_mean'),
       ('mfcc2_mean', 'spectral_centroid_mean'),
       ('mfcc2_mean', 'rolloff_mean')]
```

Q3: Analyzing Class Distribution and Addressing Imbalance

• Plot the distribution of the Genres and discuss if there is any class imbalance.

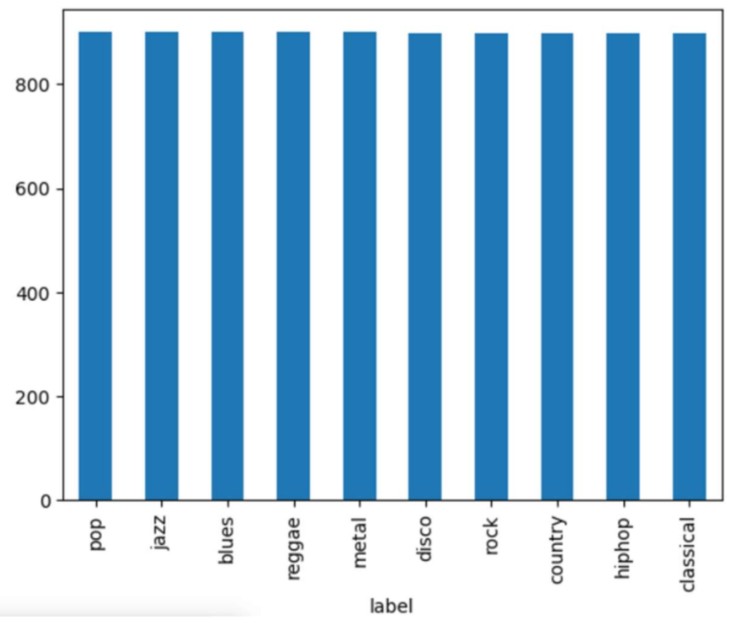
From the graph, it can be observed that there is no class imbalance, that is, the frequency of data points for each genre is same.

```
[19]: train_dataset['label'].value_counts()
```

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

```
[18]: train_dataset['label'].value_counts().plot(kind='bar')
```

```
[18]: <Axes: xlabel='label'>
```



• In case there is an imbalance, how would you solve it? List two methods.

In the case of imbalance in class distribution, balancing can be done by

1. Oversampling : We can add more copies of the classes which have lesser data points. These copies can be generated synthetically or can be augmented version od already existing data. This has to be done carefully as it may lead to overfitting.
2. Undersampling: We can also remove a few data points from classes with a lot more data than other classes. This may lead to underfitting as we may remove some important datapoints.

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

• What is overfitting, and how will you address it?

Overfitting is when our trained model has over learnt from data in the training set and hence performs very poorly when introduced with new data. When a model is overfitting, the training accuracy is very high but the validation or testing accuracy is very low.

If a model is overfitting, it can be addressed by:

1. Adding more data to the training set
2. Reducing the number of epochs
3. Selecting relevant features and removing irrelevant ones (Feature selection)
4. Reduce number of neurons from the ANN
5. Dropout – Deactivate a few neurons randomly while training the ANN

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

• What is underfitting, and how will you address it?

Underfitting is the opposite of overfitting. It is when the model has not learnt the trend or logic from the data provided to it. It is marked by very less training accuracy along with low testing/validation accuracy.

Underfitting can be addressed by:

1. Adding more data to the training set
2. Increasing number of epochs
3. Adding more features or transforming existing features for better performance
4. Adding more layers and neurons to the ANN