EE 4463, EE 5263 – Introduction to Machine Learning Project

Parth Patel	Shilpa Shilpa
yny878	ewv395
Parth.Patel@my.utsa.edu	shilpa.shilpa@my.utsa.edu

(a) Pre-processing

1. Data Loading - used Pandas to import the dataset and analyse its structure via **data.head()** and **data.tail()** (which will return the first and last 5 rows/ 13 columns of data).
2. Feature selection – columns such as Track, Artist, Year, and Label were dropped as they were not helpful in the classification task/training of the model.
3. We try to see if the data set was distributed quickly or not: we find out that there were more data available to Label zero (1169) than Label one (772). (Total data =1891). Also, we find that none of the feature columns are dependent on each other so we can't reduce the parameter for the training.

4. **Standardization (feature scaling)**: It ensures that all variables have a similar range and distribution, which is crucial for various machine learning algorithms that assume equal importance among features**.** Also, it makes features comparable and improves the performance of the learning model (SVM, Decision Tree, K-means algorithm, Naïve Bayes). There are many methods or standardizations from which we use Min-max scaling and standardScaler ().

   I. **Min-Max Scaling (MinMaxScaler)**: This technique scales data to a specified range, typically between 0 and 1. It subtracts the minimum value and divides it by the range (maximum—minimum). We use this for Naïve Bayes because it can't use the feature value with a negative values.

   X_scaled = (X — min(X)) / (max(X) — min(X))

   II. StandardScaler(): This technique transforms data to have zero(0) mean and unit(1) variance. It subtracts the mean from each data point and divides it by the standard deviation. we use this method in SVM, Decision Tree, K-means algorithm.

   Z = (X — mean(X)) / std(X)

5. **Data Splitting** – we used the in-build function of the sklearn library (train_test_split) to separate features(X) and the target variable (y) into training/testing in the ratio of 80/20 with a random state of 42.

(b) Machine–learning methods compared and your reasons for choosing them.

1. Support Vector Machines (SVM):

SVM has High-Dimensional Capability**.** SVM excels in handling data with many dimensions, pattern recognition, and making it suitable for both linear and non-linear datasets.

i. **Kernel Functions for Non-linearity**: SVM employs kernel functions to effectively deal with non-linearly separable data. These functions transform the data into higher-dimensional spaces, enabling optimal separation. We choose a **polynomial kernel** function that introduces flexibility by representing data in higher-dimensional spaces with the degree of 2.
ii. **Soft Margins**: SVM introduces soft margins to handle situations where perfect separation is not feasible. Soft margins allow for a certain degree of misclassification, and the amount of tolerance is controlled by a parameter called **C**.
    a. **C-Parameter for Regularization**: The C-parameter is crucial for preventing overfitting. It determines the balance between minimizing classification errors and maximizing the margin. A larger C value results in a smaller margin and less tolerance for misclassification, while a smaller C value allows for a larger margin and more tolerance for misclassification. So, we choose the c=11 and we get the best **accuracy of 80.2%**.

2. Decision tree:

We choose method because they are **non-parametric**, meaning they do not make assumptions about the underlying distribution of the data. This makes them useful when the data is not normally distributed or when there are outliers (we have a data set that is not normally distributed which makes it a suitable method for this project).

Hyperparameter which we choose for the Decision tree to improve accuracy:

i. **criterion**: This parameter is used to measure the quality of the split. We chose the “entropy” parameter because it calculated the measure by entropy gain.
ii. **splitter**: This parameter is used to choose the split at each node. We chose the parameter to “best” so the sub-trees can have the best split.
iii. **max_leaf_nodes**: The maximum depth of the tree. We chose the max lead nodes as 2 so it can limit the depth of three and reduce the time for processing.

By using [criterion= "entropy", splitter="best", max_leaf_nodes=2, random_state=42], we get an **accuracy of 79.41%.**

3. Naïve Bayes:

It is part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category. This approach assumes that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately. It is Very fast – no iterations are required and Works well with high-dimensional data.

Hyperparameter which we choose for the Naïve Bayes to improve accuracy.

i. **Alpha (α):** it's a smoothing parameter to handle unseen features and chose the value of alpha 1 which give use the highest **accuracy of 79.94%**.

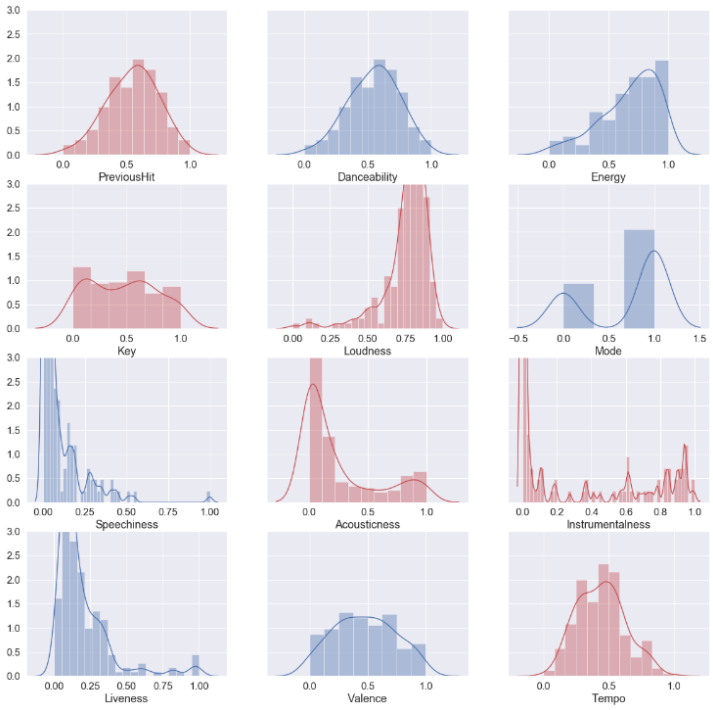
5. K-Nearest Neighbour (KNN):

We chose KNN because it tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data.

Hyperparameter of KNN:

i. **algorithm**: we choose the **kd_tree** algorithm which is a binary tree structure that recursively splits the space into half-spaces. Every node in the tree represents a d-dimensional point in space. In addition, every non-leaf node is associated with one of the d dimensions.
ii. **n_neighbors**: We tried to find the optimal K value for KNN by plotting the accuracy and error graph; we found out that k=12 gets the best **accuracy of 79.94%** and **minimum error of 0.18**.

(C) Bar plots for comparing the chosen Machine learning methods



We have drawn the normalized distribution of the graph for all the columns such as 'Danceability', 'Energy', 'Key', and 'Loudness'.

This graph help us understand the distribution of feature and which models will be more suitable to

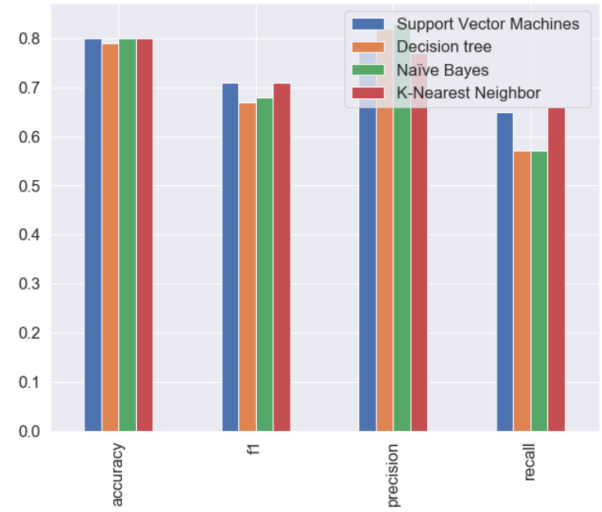


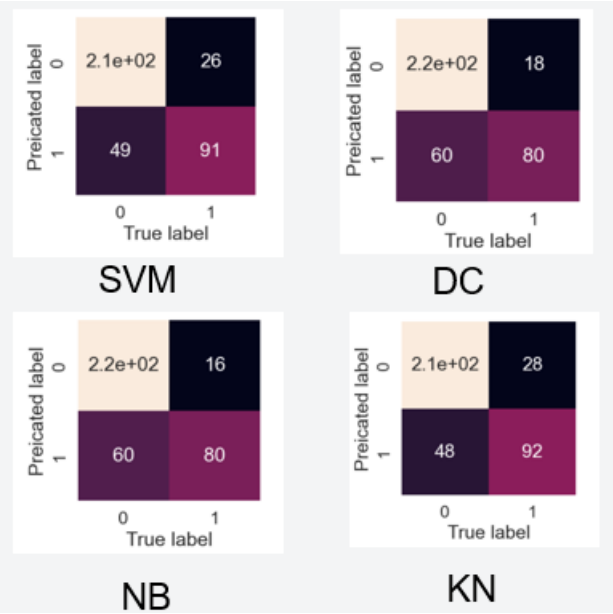
Figure 2.

	svm	Decision tree	Naïve Bayes	K-Nearest Neighbor
accuracy	0.80	0.79	0.80	0.80
f1	0.71	0.67	0.68	0.71
precision	0.78	0.82	0.83	0.77
recall	0.65	0.57	0.57	0.66

We use RandomizedSearchCV() and GridsearchCv() to maximize the output of SVM and KNN respectively. We define the parameters of both and they will return the best hyperparameter to improve the accuracy.

from only accuracy, we can't determine the best model, so we compare f1 sore, precision, and recall. We find that SVM outperforms other models.

(d) Final choice and reasons



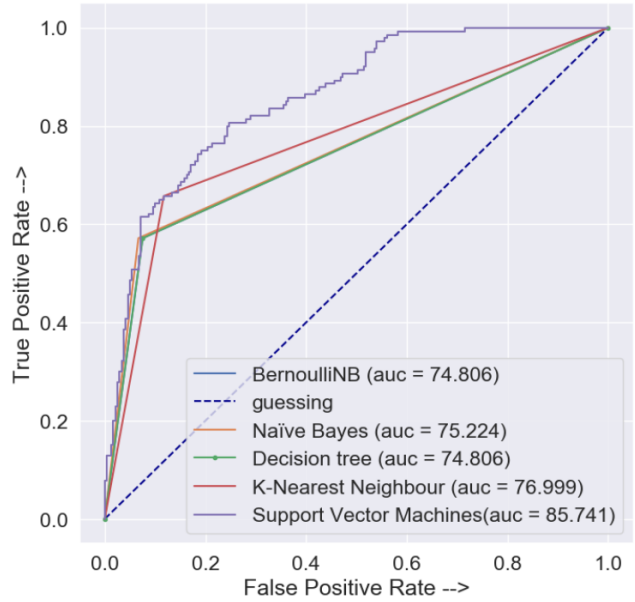
From the confusion matrix, we can see how many times we have misclassified the data as False positive and false negative.

We can see that only SVM and KNN have higher accuracy actual Label and fewer false negatives and positives.

Reason for **not** going with **KNN**:

KNN is unstable. A small change in data can lead to a vastly different decision tree. They are often relatively inaccurate and prone to overfitting. This can be rectified by replacing a single decision tree with a random forest of decision trees.

Reason we choose **SVM** as **Final MODEL**:



To Finalize the result for choosing best model we use roc cure.

ROC curve ROC curves are a comparison of a **model's true positive rate (tpr)** versus a **model's false positive rate (fpr).**

From the graph, we can see that only SVM has the highest **Area under the cure of 85.74%(AUC)**.

Using ROC, we can say that SVM outperforms the other model.

- As we can see from the graph (**Figure 2**), we get similar accuracy for all the models which is nearly 80%.

- **Precision** - Indicates the proportion of identifications (model predicted class 1) which were correct. We have max results in **NB and Decision Tree** with fewer false positive results.
- **Recall** - Indicates the proportion of actual positives that were correctly classified. We have fewer false negatives in **KNN** and **SVM.**
- **F1 score** - A combination of precision and recall: we see that **SVM** has the highest score in both.

(e) Thoughts for future work about how the model can be improved

- Experiment with adding and removing features to see whether they enhance model performance.
- Fine-tuning model hyperparameters will help to improve the accuracy.
- To improve the prediction of the model methods such as Random forests and Gradient boosting can be applied.
- Cross – Validation can help the model to improve the model's validation and performance metrics.
- Addressing the class imbalance can avoid the bias in model prediction.

(f) Contributions

- **Parth Patel** – I have worked on Support Vector Machines (SVM) and K-Nearest Neighbour (KNN), Naïve Bayes and Decision tree. Additionally, I implemented code to enhance model performance using randomsearchcv() and gridsearchcv(), resulting in the best outcomes for the projects. Specifically, I achieved an accuracy of 80.2% for SVM, 79.41% for Decision Trees, and 79.94% for both Naïve Bayes and KNN. I able to get higher accuracy and justify why I get those results, we decided to go with my code and results. we collaborated on writing the report, where I contributed relevant images.

- **Shilpa Shilpa** – I independently worked on Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naïve Bayes, and Decision Trees. My results include an accuracy of 79% for SVM, 69.92% for Decision Trees, and 64.64% for both Naïve Bayes and KNN.