DATA ANALYTICS PROJECT REPORT

CLASSIFYING SONGS PERFROMANCE ON THE BILLBOARD CHART USING ITS FEATURES

NAME:ANSHUL RAO

SRN:PES1UG20CS063

SEC:B

```
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Unnamed: 0               4560 non-null   int64  
 1   X                        4560 non-null   int64  
 2   KeySignatureConfidence   4560 non-null   float64
 3   Tempo                    4560 non-null   float64
 4   TimeSignature            4560 non-null   int64  
 5   TimeSignatureConfidence  4560 non-null   float64
 6   Year                     4560 non-null   int64  
 7   ArtistFamiliarity        4560 non-null   float64
 8   Hotness                  4560 non-null   float64
 9   end_of_fade_in           4560 non-null   float64
 10  key                      4560 non-null   int64  
 11  Loudness                 4560 non-null   float64
 12  mode                     4560 non-null   float64
 13  mode_confidence          4560 non-null   float64
 14  rank                     4560 non-null   int64  
 15  hit                      4560 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 605.6 KB
```

All these are features of the songs.

MODELS:

1. **MLR(THE WORST BECAUSE IT CANNOT USED FOR CLASSIFICATION PROBLEMS)**
2. **KNN**
3. **LOGISTIC REGRESSION**
4. **SVM**
5. **DECISION TREE**

1. INTRODUCTION:

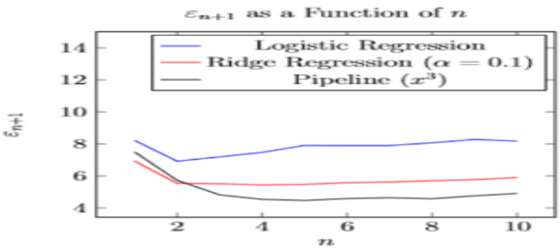
- This project offers a method for determining a song's hit potential, which can support record corporations' investment choices.
- In my project, the subset of one million songs that includes song features and the dataset for the Billboard Top 100 is employed.
- The most important attributes are selected for pre-processing the data in order to determine whether or not the music will be a hit utilizing correlation plots, heat maps, and correlation matrices.

2. INTRODUCTION TO THE CONTEXT OF THE PROBLEM

- Of course, the idea is that popular songs share a set of characteristics that make them appealing to the vast majority of the people, and that each new song can be tested against those success markers to forecast its economic potential. Every year, the music industry invests billions of dollars in new singers and songs.
- This project develops a system for measuring the hit probability of songs, which can help record labels in making investment decisions.
- ***THE MAIN GOAL IS TO CLASSIFY A SONG IF IT'S A HIT (1) OR NOT (0) BY ANALYSING ITS FEATURES***

3. PREVIOUS WORK DONE

- [1] Exploring the space of predicting the (n + 1)th position in the Billboard Hot 100 charts given the previous n positions and musical data is the purpose of this study. We can then use this forecast as the next entry to predict the (n + 2)th entry, and so on until the song falls off the charts. A variety of algorithms, including Ridge regression, logistic regression, perceptron, and pipeline, were employed to determine which generated the best results.
- It is also found that when there are more songs in a year, the songs tend to have shorter courses. These shorter courses are more straightforward in their portrayal of general patterns. When Gracenote features, such as Mood and WeeksInChart, were added, they just crowded the same classiers that would provide superior results without them. In this example, pipeline with n = 4 was the best performing algorithm. This graph has these qualities.



- [2]They investigate artificial music analysis to identify likely popular tunes. They gather acoustic and lyric information from each song and use common classifiers, specifically Support Vector Machines and boosting classifiers, to differentiate hits from non-hits.
- Their findings indicate that there is a distinct thread connecting hit tunes. The results show that lyric-based features are marginally more effective than audio-based features at differentiating hits for the characteristics utilised.
- Combining features does not enhance performance considerably. Figures 1 and 2 illustrate the ROC area averaged over the experimental database's ten cross validation cuts for SVM and boosting classifiers trained on acoustic and lyrics-based features.
- They also experiment with merging acoustic and lyric-based elements. This was accomplished by concatenating the vectors for the two representations. Figure 3 depicts the findings of this experiment.

1

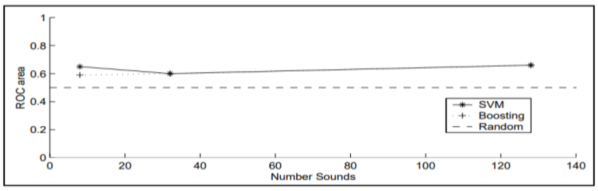
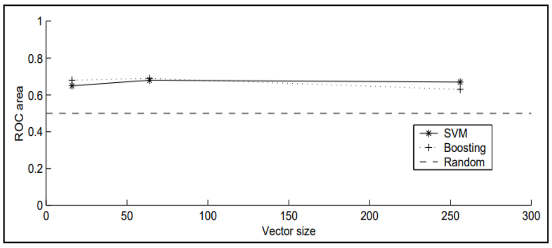
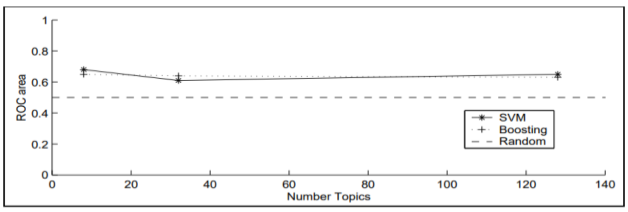


Figure 1: Average ROC area for acoustic-based features with various numbers of sounds for SVM and boosting classifiers



2 Figure 3: Average ROC area for combined acoustic and lyrics features with varying vector sizes for SVM and boosting classifiers



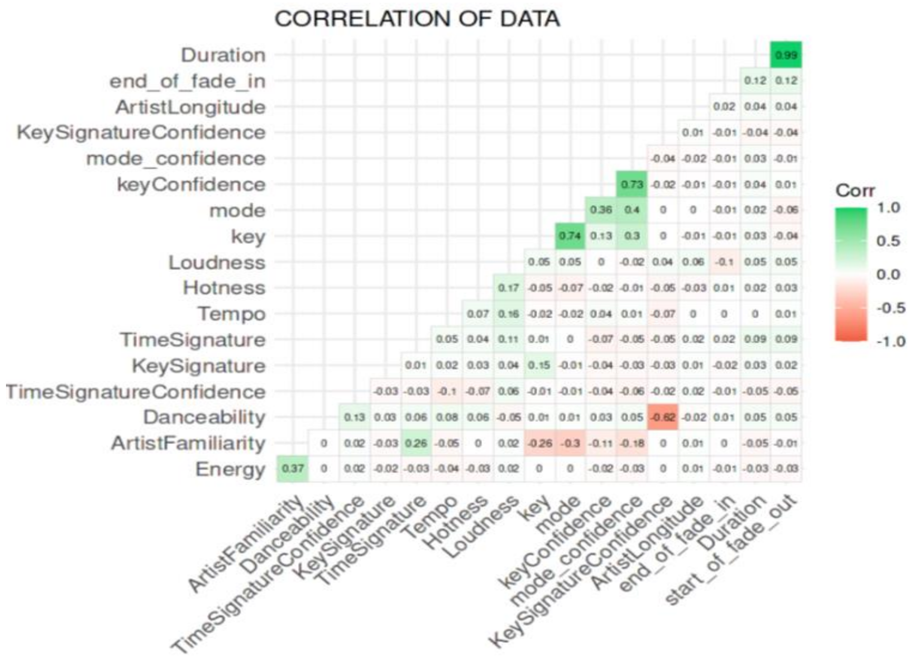
3 Figure 2: Average ROC area for lyric-based features with various numbers of topics for SVM and boosting classifiers

4. DATA RESOURCES

- my project analyses the song's features and predicts whether or not it will be a hit.
- Data was gathered from the one million data set and the top 100 songs on the Billboard charts.

- The Million Song Dataset is a collection of audio characteristics and metadata for a million modern popular music tracks that is freely available.
- The basis of the dataset is The Echo Nest's feature analysis and metadata for one million songs.
- The billboard top 100 song dataset, together with their accompanying ranks, is used to match songs in the million song dataset.
- I am using a combined dataset

5.Data preprocessing



- Correlated attributes are detected and deleted because their behaviour and influence in prediction calculations are comparable, making preserving attributes with similar impacts superfluous. The attributes with strong correlation, such as key signature and key, key confidence and mode confidence, start of fade out and length, were identified using the correlation matrix, heat map, and correlation plot.
- A heat map and correlation plot are used to better see and comprehend the relationship between characteristics. In R, the heatmap and corrplot functions are used to accomplish this.
- Because it contained only 0 values, qualities like energy and danceability were eliminated. Numeric properties such as duration, key confidence, key signature, and fade out start time are removed. The heat maps and correlation plot are provided below.
- The correlation matrix is obtained by using the functions from the ggcorrplot library.

```python
data = data.drop(labels="Title", axis=1)
data = data.drop(labels=1474, axis=0)
data = data.drop(labels=1476, axis=0)

data['key'] = pd.to_numeric(data['key'])
data['Tempo'] = pd.to_numeric(data['Tempo'])
data['Hotness'] = pd.to_numeric(data['Hotness'])
```

Removing inconsistent rows and converting object columns to numeric data type.

5. Models:

1.Multiple Linear Regression

- The most prevalent type of linear regression analysis is multiple linear regression. Multiple linear regression is a predictive approach that is used to explain the relationship between one continuous dependent variable and two or more independent variables.
- Songs which lie in the top 50 of the billboard chart are considered hit (have the values 1) and the rest have the value 0.4 In multiple linear regression,
- the **hit** column is the dependant variable and the song features such as KeySignatureConfidence , TimeSignature ,TimeSignatureConfidence ,ArtistFamiliarity , Hotness , end-of-fade-in , key, Loudness , mode, mode-confidence are considered as independant variable
- Train and test sets are constructed to train the model and validate its accuracy.
- The function ln in **e1071** is used to build the multiple linear regression model in R, and the **predict function** is used to predict the test data values.
- Using the confusion matrix function in the caret package, the accuracy, precision, and specificity of the mode are determined. The function accuracy is also used to determine the model's accuracy. The confusion matrix and the results of the multiple linear regression model are shown below.

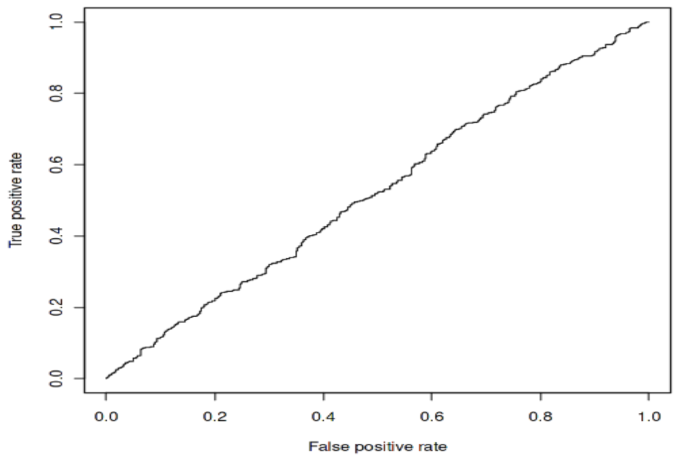
```
Confusion Matrix and Statistics

     0   1
 0 503 559
 1   0   0

               Accuracy : 0.4736
                 95% CI : (0.4432, 0.5042)
    No Information Rate : 0.5264
    P-Value [Acc > NIR] : 0.9997

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.4736
         Neg Pred Value :    NaN
             Prevalence : 0.4736
         Detection Rate : 0.4736
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

- The accuracy is around 50% (47.63%) ☹

- AIC=5081.67743123704
- BIC=5217.20254606913

THE ROC CURVE

```
<bytecode: 0x55a665b6fa30>
<environment: namespace:pROC>
```



A ROC curve on a 45 degree indicates **a worthless model**.

- All the linear regression methods, have a major drawback that most systems are not linear in reality. Linear models attempt to fit a line through one dimensional, a plane through two dimensional, and a generalization of a plane through higher dimensional data sets.

  Let's try a better model!!

  This is a binary classification problem thus regression models are useless.

Thus we use classification models:

1.knn

2.logistic reg

3.decision tree

4.svm

1.KNN:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[590,  86],
       [ 58, 634]])
```

```
 [ 58 634]]
Classification report

              precision    recall  f1-score   support

           0       0.91      0.87      0.89       676
           1       0.88      0.92      0.90       692

    accuracy                           0.89      1368
   macro avg       0.90      0.89      0.89      1368
weighted avg       0.90      0.89      0.89      1368
```

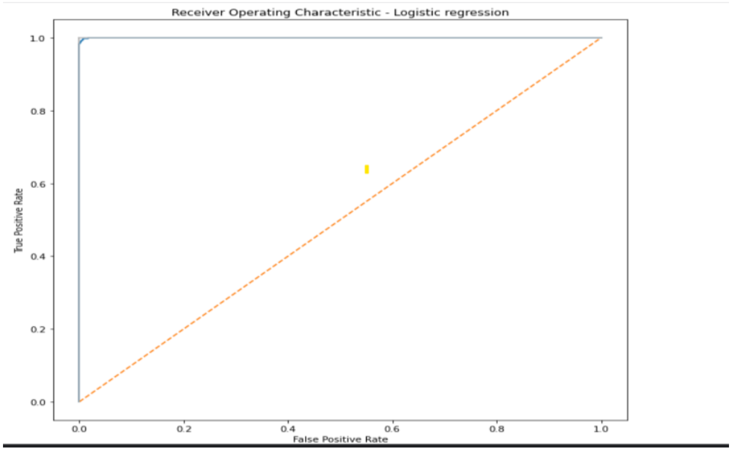
Accuracy=0.89

2.LOGISTIC REGRESSION:

```
from sklearn.linear_model import LogisticRegression
classifier1=LogisticRegression(random_state=0)
classifier1.fit(X_train,y_train)
y_pred=classifier1.predict(X_test)
acc2=accuracy_score(y_test,y_pred)
```

```
acc2
```

```
0.9956140350877193
```



THE ROC CURVE SHOWS THAT THIS IS A GOOD MODEL
THE CURVE IS VER FAR FROM THE 45 DEGREE LINE

ACCURACY=0.9956

3.DECISION TREE:

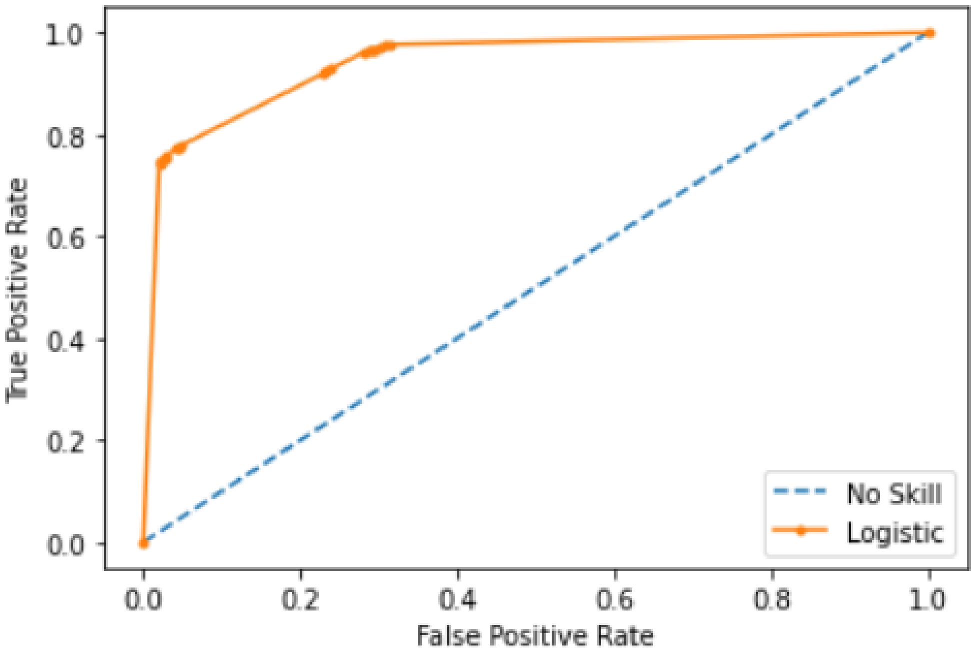
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV

tree_clf = DecisionTreeClassifier()
tree_clf.fit(X_train,y_train)
tree_clf_gs = RandomizedSearchCV(tree_clf, parameters)
tree_clf_gs.fit(X_train,y_train)
```

```
y_pred = tree_clf.predict(X_test)
acc3=accuracy_score(y_test, y_pred)
acc3
```

```
0.8823099415204678
```

```
 [ 94 598]]
Classification report

              precision    recall  f1-score   support

           0       0.87      0.90      0.88       676
           1       0.90      0.86      0.88       692

    accuracy                           0.88      1368
   macro avg       0.88      0.88      0.88      1368
weighted avg       0.88      0.88      0.88      1368
```

ACCURACY=0.8823



ROC CURVE SHOWS THE MODEL IS GOOD

4.SVM:

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
acc4=accuracy_score(y_test, y_pred)
acc4
```

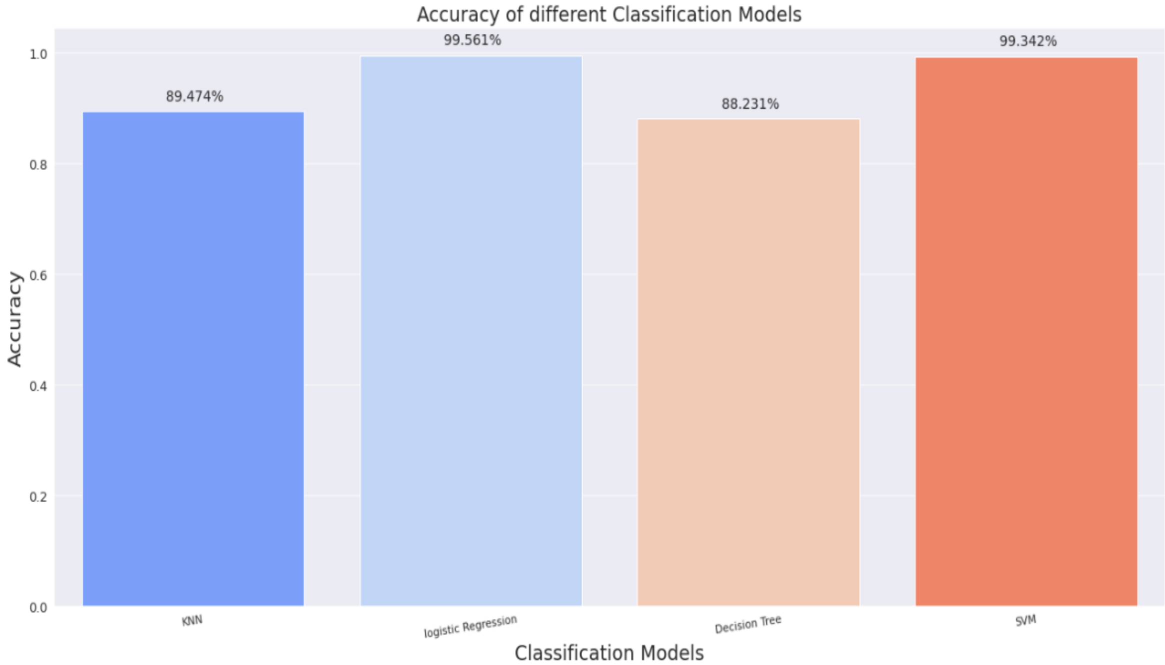
```
0.993421052631579
```

ACCURACY:0.99342

RESULT:

```
mylist=[]
mylist2=[]
mylist.append(acc1)
mylist2.append("KNN")
mylist.append (acc2)
mylist2.append( "logistic Regression")
mylist.append(acc3)
mylist2.append("Decision Tree")
mylist.append(acc4)
mylist2.append( "SVM")

plt. rcParams ['figure.figsize']=22, 10
sns.set_style("darkgrid")
ax = sns.barplot(x=mylist2, y=mylist, palette = "coolwarm", saturation =1.5)
plt.xlabel( "Classification Models", fontsize = 20 )
plt.ylabel("Accuracy", fontsize = 20)
plt.title("Accuracy of different Classification Models", fontsize = 20)
plt.xticks(fontsize = 11, horizontalalignment = 'center', rotation = 8)
plt.yticks(fontsize = 13)
for p in ax.patches:
    width, height = p.get_width(), p.get_height ()
    x, y = p.get_xy ()
    ax.annotate (f' {height:.3%}', (x + width/2, y + height*1.02), ha='center', fontsize = 'x-large')
plt.show()
```

PLOT TO SHOW ACCURACY OF DIFFERENT CLASSIFICATION MODELS:



FROM THE GRAPH WE CAN SEE THE SVM MODEL GIVES THE BEST PERFROMANCE
THE ORDER IS :
1.SVM
2.LOGISTIC REGRESSION
3.KNN
4.DECISION TREE