Spotify Song Hit Prediction Using Machine Learning

Stefan Bukarica -190563930

Duc Minh Nguyen - 203009140

The Problem

Preprocess our Dataset

Train multiple models

Select the best performing models

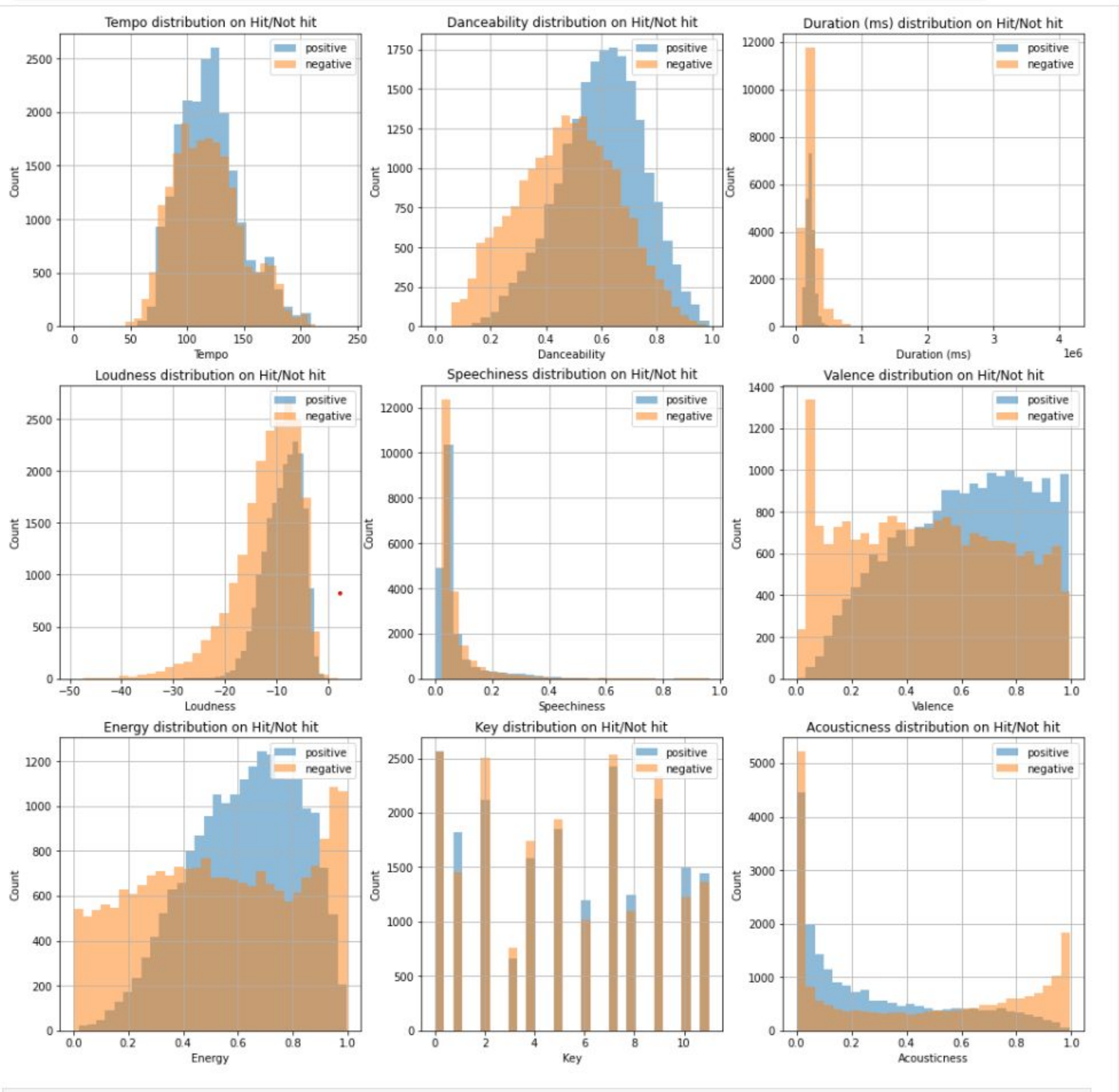
Apply predictions and test to see if we can properly classify hit songs.

Dataset

- Our dataset features 19 Attributes
- We add one to indicate what decade songs are from named “decade”
- Track, Artist, URI, Danceability, Energy, Key, Loudness, Mode, Speechiness, Acousticness, Instrumentalness, Liveliness, Valence, Tempo, Duration (in ms), Time signature, Chorus Hit, Sections, Target, and Decade

Attribute Analysis

- Now we want to explore what ranges for each attribute contribute to making a hit.
- We designed this set of graphs, the blue corresponds to the songs that are a hit



Key Hit Ranges

- We can see now that the following ranges of each attribute likely produce a hit (we will attempt to test this later in our machine learning model)

Attribute	"Hit" Range
Tempo	80, 160
Danceability	0.5, 1.0
Loudness	-20 , 0
Speechiness	0.0 , 0.2
Valence	0.5, 1.0
Energy	0.4 , 0.9
Key	2,4,6,8
Acousticness	0.0 , 0.6

Preprocessing

- We remove the categorical attributes as they have nothing to do with training our model
- They are just identifiers for the individual songs
- We remove
  - Track (name of song)
  - Artist (song creators name)
  - URI (spotify api link)

Preprocessing (continued)

- Next we have to standardize the dataset.
- We do this since different attributes are measured at different scales
- This results in an unequal contribution to model fitting, and more importantly results in **Bias**

Before Training

- With preprocessing, we combine the train/test split to be 80/20 and apply it to our dataset before training as shown on the right:

```
def preprocess(df):
    dfCopy = df.copy()

    #we want to drop categorical values that have nothing to do with our analysis, track name, artist name, and uri (link from spotify api)
    #there are too many elements in these columns we would have to set up too many dummy variables thus making a df with too many cols
    dfCopy = dfCopy.drop(["track", "artist", "uri"], axis = 1)

    #since we predict target (hit or not) we split it

    y = dfCopy["target"]
    x = dfCopy.drop("target", axis = 1)

    #training and testing
    #higher training % = more accuracy, common practice is to use 70/30
    # due to size of our dataset (small) we will use 80/20
    x_train,  x_test, y_train, y_test = train_test_split(x,y, train_size = 0.8, shuffle = True, random_state = 1)

    #scale values to make them closer together

    scale = StandardScaler()
    scale.fit(x_train)
    x_train = pd.DataFrame(scale.transform(x_train), index = x_train.index, columns = x_train.columns)
    x_test = pd.DataFrame(scale.transform(x_test), index = x_test.index, columns = x_test.columns)

    return x_train, x_test, y_train, y_test
```

Model Selection

- After preprocessing our data using sk-learn's StandardScaler, we begin testing a series of ML models followed by their accuracy.

```
                 Logistic Regression trained.
                 K-Nearest Neighbors trained.
                       Decision Tree trained.
Support Vector Machine (Linear Kernel) trained.
   Support Vector Machine (RBF Kernel) trained.
                      Neural Network trained.
                 Logistic Regression: 74.13%
                 K-Nearest Neighbors: 75.27%
                       Decision Tree: 72.73%
Support Vector Machine (Linear Kernel): 74.17%
   Support Vector Machine (RBF Kernel): 80.15%
                      Neural Network: 79.94%
```

Model Selection (cont'd)

- We will take the two highest accuracy models to use in our prediction.
- Support Vector Machine (RBF Kernel)
- Neural Network

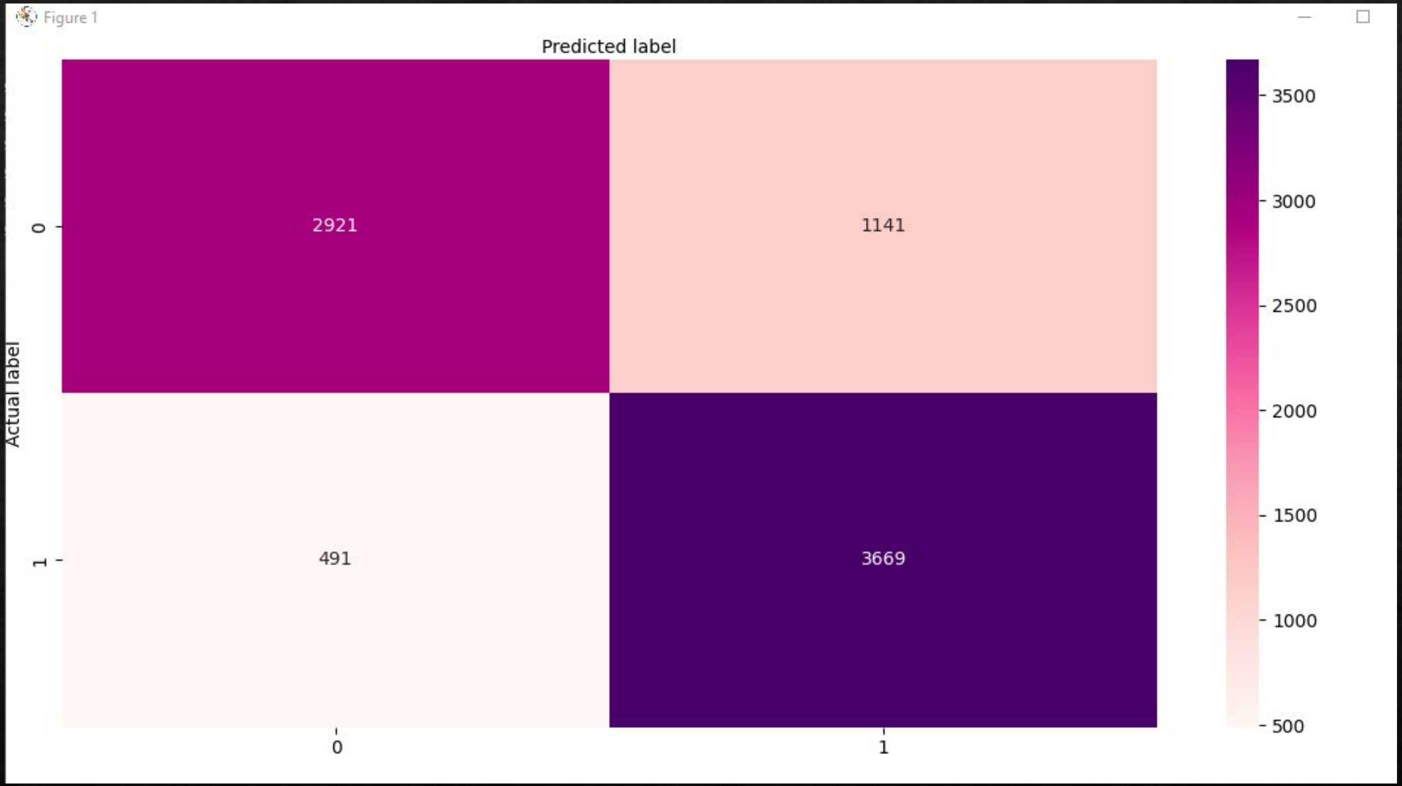
Prediction

- Our first step was to create a set of predictions with both the SVM model and the Neural Network Model
- We then want to create a confusion matrix to compare these predictions to the test values (actual values)

```
svcModel = SVC()
NNmodel = MLPClassifier()

svcModel.fit(x_train, y_train)
NNmodel.fit(x_train, y_train)

y1_pred = svcModel.predict(x_test)
y2_pred = NNmodel.predict(x_test)

#calculate a confusion matrix

cnf_matrix = metrics.confusion_matrix(y_test, y1_pred)
cnf2_matrix = metrics.confusion_matrix(y_test, y2_pred)
```

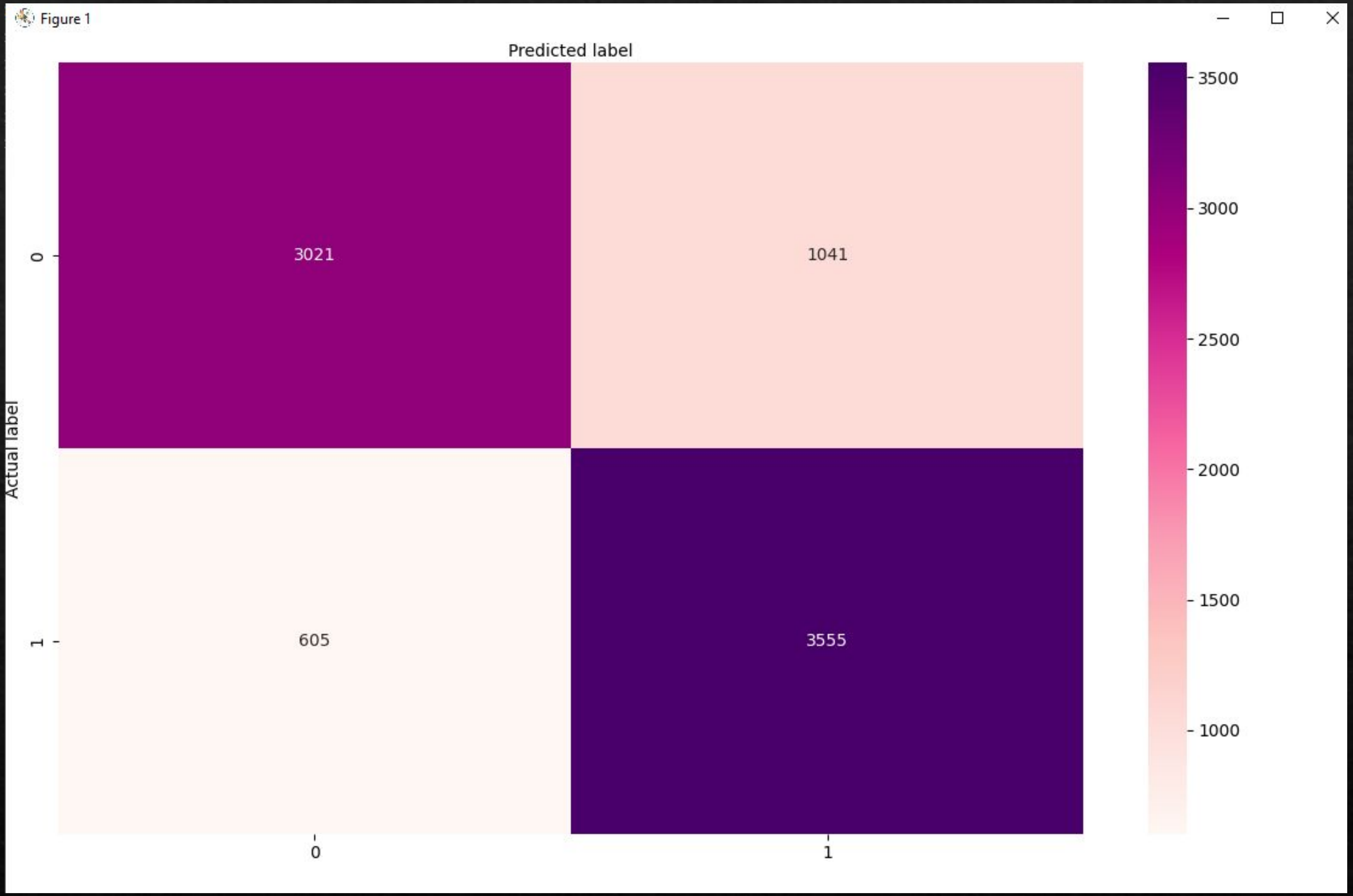
Prediction (continued)

- This is the Confusion Matrix for the SVM model, the darker colours representing the overlap of correct predictions.



Prediction (Continued)

- This is the Confusion Matrix for the Neural Network model, the darker colours representing the overlap of correct predictions.



Conclusion

- We are able to effectively predict both hit songs and flop songs with an SVM model and Neural Network model.
- A larger dataset could have garnered better results
- A further investigation should be done to expand what we consider a "hit" song to begin with

Thank you for watching