Predicting song popularity

170183149

4th December 2019

1 Introduction

The Million Song Dataset [1] is a dataset containing features of one million contemporary popular songs from analysis provided by The Echo Nest [2]. Because the dataset is so large it is only appropriate to work with a limited number of songs and features so this report will use a reduced size dataset with 50000 songs and 11 features: duration, fade_in, fade_out, loudness, mode, tempo, time_sig, year, artist_fam, artist_pop, song_pop. fade_in and fade_out are the number of seconds from the start of the track that the music begins and begins to fade respectively while artist_fam and artist_pop give numerical estimates of an artists familiarity and popularity to the world. The variable song_pop shows the popularity of the song, 0 for unpopular, 1 for popular, and the aim of this report is to find the method that will most accurately predict which of these two classes the song will fall into based on its other features. Methods discussed will be logistic regression, discriminant analysis, decision trees, and neural networks.

2 Logistic Regression

One way to sort the observation into classes is to implement the Logistic Regression Algorithm, which estimates the probability of a data point being in a certain class using the Sigmoid-Function,

$$p(t) = \frac{e^t}{1 + e^t}$$

, t being the linear model. A disadvantage of this is that it separates the observations with a linear boundary so requires the data to be linearly separable which is not often the case. Applying this algorithm to every input and comparing the output to the values in the song_pop column gives an idea of how accurate this method is. Creating a model based on one dataset means there is a risk of overfitting the data to that particular dataset and the analysis not being as applicable to further data. Therefore a better idea of the accuracy of the model is found using cross-validation, which creates a model based on a subset of the data which is then used to predict the outcome for the rest of

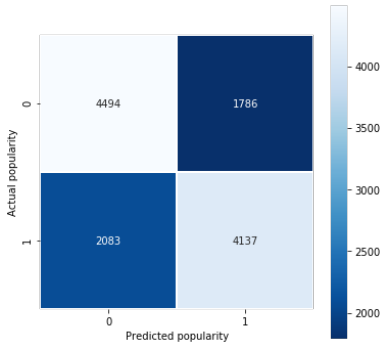

Figure 1: Correlation matrix

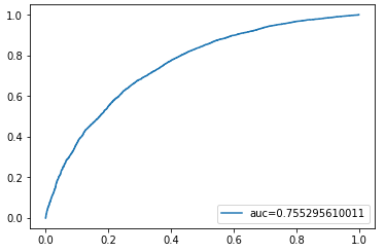

Figure 2: ROC

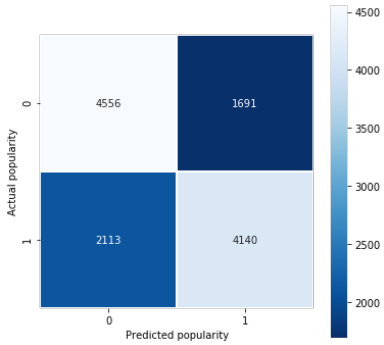

Figure 3: Correlation matrix

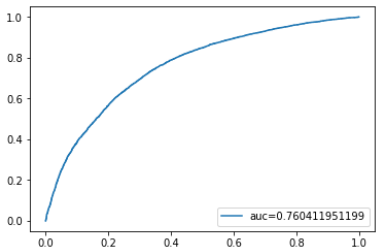

Figure 4: ROC

the data. Figure 1 is the correlation matrix which shows how many data points the model predicted correctly and how many it didn't within a subset of 12500 observations. Figure 2 shows the ROC (reciever operating characteristic) curve which is a plot of sensitivity (the proportion of all 1s that are actually predicted to be 1s) against specificity (proportion of all 0s that are actually predicted to be 0s). The area under this curve is also a good measure of the accuracy of the model and an area closer to 1 indicates a good model.

3 Discriminant Analysis

Looking at plots of the variable song_pop against the other features it looks like both classes are similar in variance therefore it would be appropriate to use the method of Linear Discriminant Analysis (LDA) to try and classify the data. LDA assumes that the data is based on a random samples from a multivariate normal distribution therefore it models each class by the normal distribution and finds the means and variance which maximise the likelihood of the data. Using

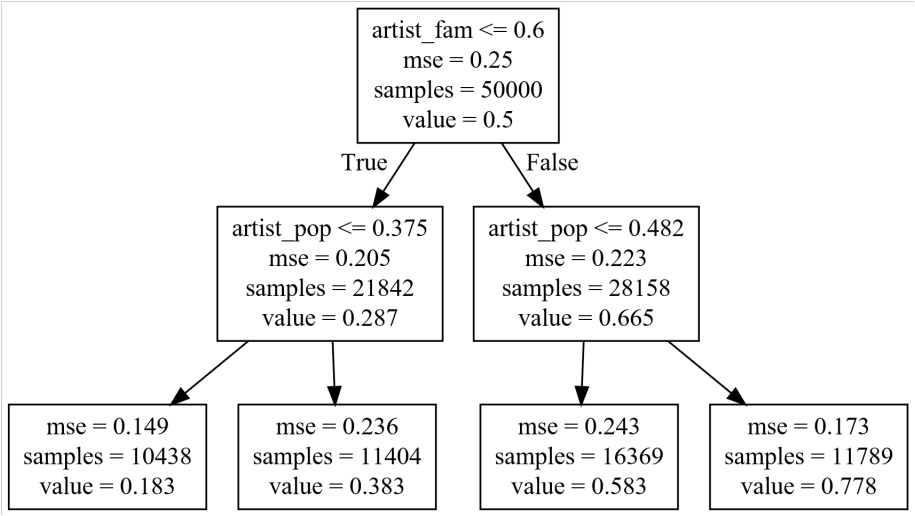

Figure 5: Decision Tree

cross-validation again to create a more useful model the correlation matrix and ROC curves are shown in Figures 3 and 4 respectively.

4 Decision Trees

Another approach to solving a classification problem is to form a decision tree by starting with the complete dataset at the root node and choosing a variable and threshold, splitting a branch into two 'child' nodes that contain data with values either side of the threshold. We stop adding layers to the tree when every observation is correctly classified. 50 trees were used to classify this data which was a value decided on by measuring the mean squares error of the model and trying to find the right number of trees so that the mean squares error of a training set was close to or below that of the model. The errors for 40 and 60 trees both exceeded the error for 50 trees which was 0.431908768144 which showed that those models would be underfitting and overfitting the data respectively. Figure 5 shows the first two layers of the tree produced, with the features, thresholds and errors shown.

5 Neural Networks

This method of classification involves using a network of weighted nodes (their own linear models) to simulate a function which returns an output of a probability of either 0 or 1 from an input. Having a simple network works best for

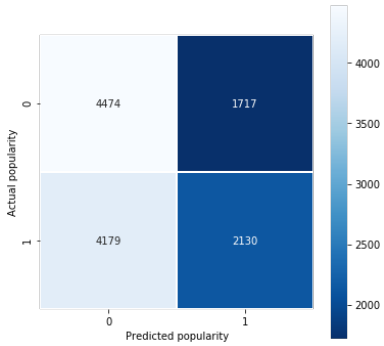

Figure 6: Correlation matrix

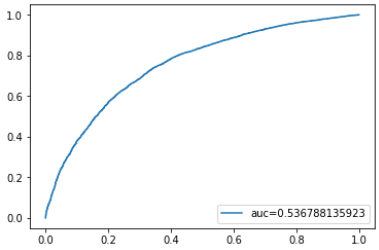

Figure 7: ROC

this data as it is very easily overfitted, the chosen dimensions being [3,2]. The correlation matrix and ROC curves are shown in Figures 6 and 7 respectively.

6 Analysis

Method	Accuracy	Accuracy of test set on full set	AUC
Logistic Regression	0.691173823476	0.68896	0.755045966563
Discriminant Analysis	0.693793875878	0.69464	0.760411951199
Decision Trees	0.9999	0.71264	0.706136609487
Neural Networks	0.52764	0.52888	0.532947975547

The accuracy score of the method alone isn't indicative of which is the best method of classification. In the case of logistic regression, the accuracy is shown to be reduced when applied to the larger dataset and with a decision tree there is almost full accuracy which means that the model has been overfitted to the data and will be less accurate given unused data as shown in the accuracy score for the test set. Neural networks have low scores on all indicators which shows that it is too complex a method to be applied in this analysis as there are too few variables for it to be necessary. The best indicator of a good model is how well the model fits to new data so although discriminant analysis produces a better ROC score, using decision trees would be more appropriate as the accuracy rate for the test set will be similar to the accuracy rate on any new data.

7 Appendix

References

[1] http://millionsongdataset.com/

[2] http://the.echonest.com/

```
'''Import packages and data'''

%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import export_graphviz
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

songs = pd.read_csv('C:/Users/charl/OneDrive/Documents/Machine Learning
/Project 2/songstrain.csv')


'''Logistic Regression'''

X = songs.iloc[:,0:10]
y = songs.song_pop
logreg=LogisticRegression()
logreg.fit(X,y)
y_pred=logreg.predict(X)
score = logreg.score(X,y)
print(score)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
logreg=LogisticRegression()
logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
cm=metrics.confusion_matrix(y_test,y_pred)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
plt.figure(figsize=(6,6))
sns.heatmap(cm,annot=True,fmt="d",linewidths=.5,square=True,cmap="Blues_r")
plt.ylabel("Actual popularity")
plt.xlabel("Predicted popularity")
plt.show()
y_pred_proba=logreg.predict_proba(X_test)[::,1]
fpr,tpr,_=metrics.roc_curve(y_test,y_pred_proba)
auc=metrics.roc_auc_score(y_test,y_pred_proba)
print(auc)
```

```
plt.plot(fpr,tpr,label="auc="+str(auc))
plt.legend(loc=4)
plt.show()

'''Discriminant Analysis'''

X = X.values
y = y.values
lda=LDA()
lda.fit(X,y)
lda.scalings_
lda.explained_variance_ratio_
y_pred=lda.fit(X,y).predict(X)
print(lda.score(X,y))
cm=metrics.confusion_matrix(y,y_pred)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
y_test_pred=lda.fit(X_train,y_train).predict(X_test)
y_predprob=lda.predict_proba(X_test)[::,1]
cm=metrics.confusion_matrix(y_test,y_test_pred)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
plt.figure(figsize=(6,6))
sns.heatmap(cm,annot=True,fmt="d",linewidths=.5,square=True,cmap="Blues_r")
plt.ylabel("Actual popularity")
plt.xlabel("Predicted popularity")
plt.show()
fpr,tpr,_=metrics.roc_curve(y_test,y_predprob)
auc=metrics.roc_auc_score(y_test,y_predprob)
print(auc)
plt.plot(fpr,tpr,label="auc="+str(auc))
plt.legend(loc=4)
plt.show()

'''Decision trees'''

X = songs.iloc[:,0:10]
y = songs.song_pop
treereg=DecisionTreeRegressor(max_depth=2)
treereg.fit(X,y)
treereg.feature_importances_
features = ["duration", "fade_in", "fade_out", "loudness", "mode",
"tempo", "time_sig", "year", "artist_fam","artist_pop"]
export_graphviz(treereg,out_file="tree.dot",feature_names=features)
rfreg=RandomForestRegressor(n_estimators=50)
```

```
rfreg.fit(X,y)
yhat=treereg.predict(X)
yhat
print(np.sqrt(metrics.mean_squared_error(yhat,y)))
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
rfreg.fit(X_train,y_train)
yhat=rfreg.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(yhat,y_test)))
y1=y
rfclf=RandomForestClassifier(n_estimators=50)
rfclf.fit(X,y1)
yhat1=rfclf.predict(X)
cm=metrics.confusion_matrix(y1,yhat1)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
X_train,X_test,y1_train,y1_test=train_test_split(X,y1,test_size=0.25)
rfclf.fit(X_train,y1_train)
yhat1=rfclf.predict(X_test)
cm=metrics.confusion_matrix(y1_test,yhat1)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
auc=metrics.roc_auc_score(y1_test,yhat1)
print(auc)


'''Neural Networks'''

X = X.values
y = y.values
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(7, 2), random_state=1, solver='lbfgs')
y_pred = clf.predict(X)
cm=metrics.confusion_matrix(y,y_pred)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
y_pred = clf.predict(X_test)
cm=metrics.confusion_matrix(y_test,y_pred)
accuracy=(cm[0,0]+cm[1,1])/(cm[0,0]+cm[0,1]+cm[1,0]+cm[1,1])
print(accuracy)
y_predprob=clf.predict_proba(X_test)[::,1]
auc=metrics.roc_auc_score(y_test,y_predprob)
print(auc)
```