Popularity Prediction of Spotify Music

By Nick Seguljic

The Problem

Identify features and create models to predict the popularity of a song based on the features provided by the Spotify API

Overview

- After model development the Linear Regression was best at predicting popularity with an MAE of 6.26 and RMSE of 8.65

- Random Forest Model was best at making the binary prediction if a song would be popular not with an accuracy of 90% and MAE of 1.64 and RMSE of 0.32

	Model	Accuracy	MAE	MSE	RMSE
2	KNeighborsClassifier	0.000000	37.506617	1526.300102	39.067891
3	DecisionTreeClassifier	0.011266	19.802104	540.345029	23.245323
4	AdaBoostClassifier	0.036037	11.390024	236.521208	15.379246
1	RandomForestClassifier	1.000000	0.000000	0.000000	0.000000
0	LinearRegression	NaN	6.256225	74.748668	8.645731

The Data and Feature Engineering

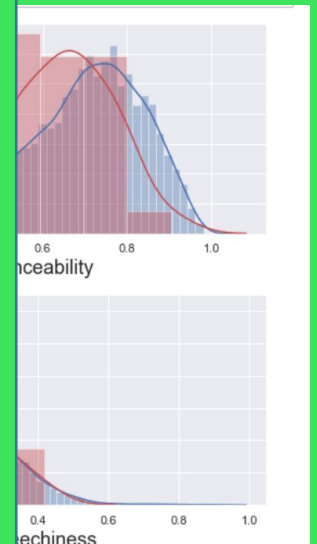
In [2]:
```
rawdata = pd.read_csv('SpotifyFeatures.csv')
print(rawdata.head())
print(rawdata.tail())
rawdata.shape
rawdata.describe()
```

```
   genre      artist_name                                    track_name  \
0  Opera   Giuseppe Verdi  Stiffelio, Act III: Ei fugge! … Lina, pensai c...
1  Opera  Giacomo Puccini  Madama Butterfly / Act 1: ... E soffitto e pareti
2  Opera  Giacomo Puccini      Turandot / Act 2: Gloria, gloria, o vincitore
3  Opera   Giuseppe Verdi       Rigoletto, Act IV: Venti scudi hai tu detto?
4  Opera   Giuseppe Verdi           Don Carlo / Act 4: "Ella giammai m'amò!"

                 track_id  popularity  acousticness  danceability  \
0  7EsKYeHtTc4H4xWiTqSVZA          21
1  7MfmRBvqaW0I6UTxXnad8p          18
2  7pBo1GDhIysyUMFXiDVoON          10
3  02mvYZX5aKNzdqEo6jF20m          17
4  03TW0jwGMGhUabAjOpB1T9          19

   duration_ms    energy  instrumentalness ke
0       490867  0.23100          0.000431  C
1       176797  0.20100          0.028000  D
2       266184  0.47000          0.020400
3       288573  0.00605          0.000000
4       629760  0.05800          0.146000

   speechiness    tempo time_signature  vale
0       0.0547   86.001            4/4   0.0
1       0.0581  131.798            4/4   0.3
2       0.0383   75.126            3/4   0.0
3       0.0480   76.493            4/4   0.0
4       0.0493  172.935            4/4   0.0
```

In [5]:
```
duplicateRowsDF = rawdata[rawdata.duplicated()]
duplicateRowsDF

bins = np.array([100000,300000,500000,700000,900000,1000000])
rawdata["bucket"] = pd.cut(rawdata.duration_ms, bins)

rawdata['lognorm_duration'] = np.log(rawdata['duration_ms'])

rawdata['Count'] = \
    rawdata.groupby('artist_name', as_index=False)['artist_name'].transform(lambda s: s.count())

rawdata['popular'] = np.where(rawdata['popularity'] >= 40, 'popular', 'not-popular')
print (rawdata)
```

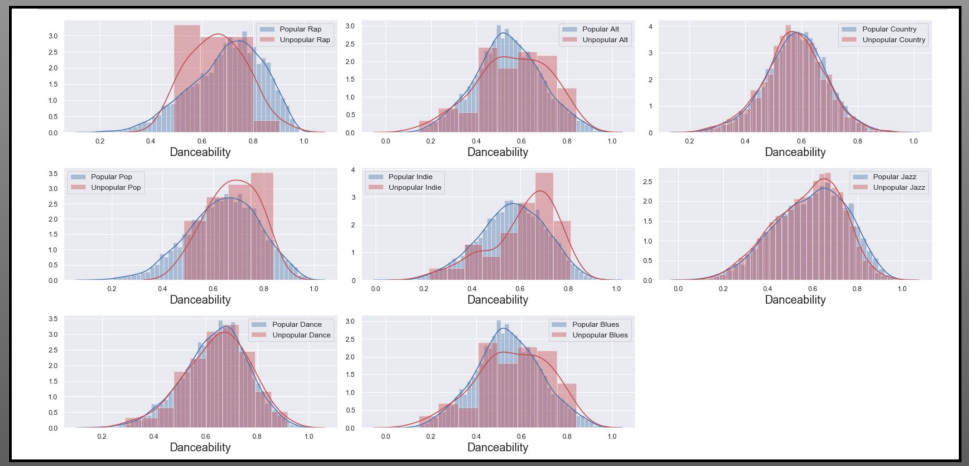
Data Analysis

	Popularity Correlation
Unnamed: 0	0.797325
popularity	1.000000
acousticness	0.008367
danceability	0.059371
duration_ms	0.022568
energy	0.045087
instrumentalness	0.007961
liveness	0.055524
loudness	0.055885
speechiness	0.084576
tempo	0.017993
valence	0.003175
lognorm_duration	0.017973
Count	0.127838



Data Analysis

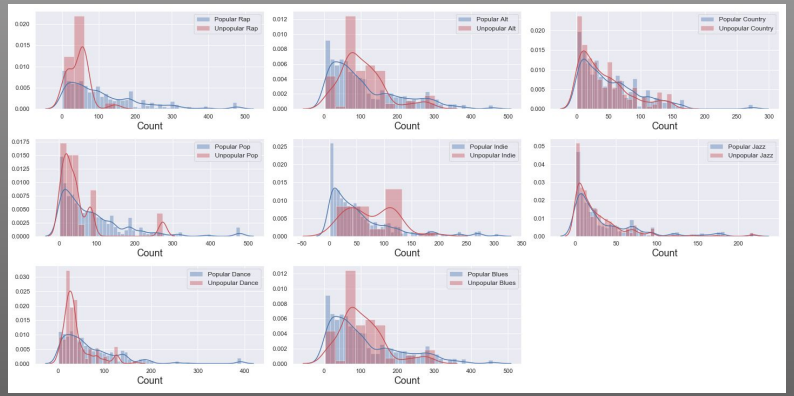
Are popular and unpopular data sets statistically different?




```
Rap: "+ Ttest_indResult(statistic=-20.68151316779942, pvalue=6.363020082246064e-93)
Alternative: "+ Ttest_indResult(statistic=-41.18452352839036, pvalue=0.0)
Country: "+ Ttest_indResult(statistic=-74.65733407824712, pvalue=0.0)
Pop: "+ Ttest_indResult(statistic=-29.62200817454119, pvalue=1.9938456189947248e-184)
Ind: "+ Ttest_indResult(statistic=-27.148616387485827, pvalue=2.0741653447726512e-156)
Jazz: "+ Ttest_indResult(statistic=-100.24883209714802, pvalue=0.0)
Dance: "+ Ttest_indResult(statistic=-68.71157367452614, pvalue=0.0)
Blues: "+ Ttest_indResult(statistic=-41.18452352839036, pvalue=0.0)
```

Data Analysis

Are popular and unpopular data sets statistically different?




```
Rap: "+ Ttest_indResult(statistic=-3.3050728333663453, pvalue=0.0009531375635210421)
Alternative: "+ Ttest_indResult(statistic=-0.7445615013922593, pvalue=0.4565555593879941)
Country: "+ Ttest_indResult(statistic=-9.00727506951008, pvalue=2.563196871401885e-19)
Pop: "+ Ttest_indResult(statistic=-2.133484730581887, pvalue=0.0329108441834335)
Ind: "+ Ttest_indResult(statistic=2.4964435087149073, pvalue=0.012561301843841095)
Jazz: "+ Ttest_indResult(statistic=-12.163873541101108, pvalue=8.659592109589175e-34)
Dance: "+ Ttest_indResult(statistic=-5.717353099471869, pvalue=1.1176843660026419e-08)
Blues: "+ Ttest_indResult(statistic=-0.7445615013922593, pvalue=0.4565555593879941)
```

From this we can reject the null hypothesis for Jazz, and unlike before we can accept the null hypthesis for Dance and Pop. This makes sense because there was a clean distinction for popular versus unpopular Jazz music, and the plots generated for Dance and Pop looked identical.

Models

Popularity Score Prediction

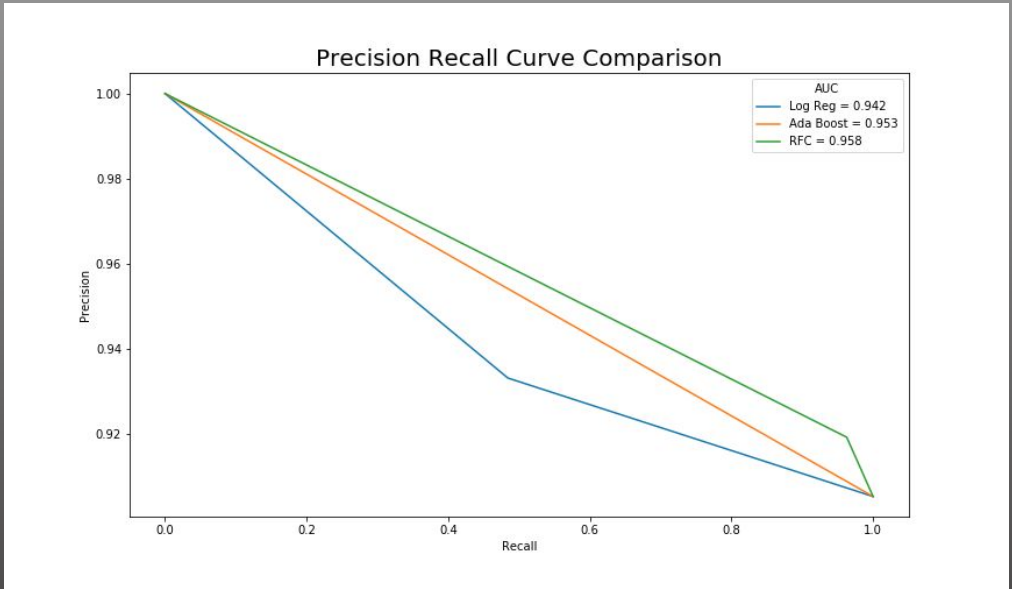
	Model	Accuracy	MAE	MSE	RMSE
4	AdaBoostClassifier	0.000068	44.764981	2149.432372	46.361971
2	KNeighborsClassifier	0.000271	36.673023	1462.290126	38.239902
3	DecisionTreeClassifier	0.016491	16.878181	409.673295	20.240388
1	RandomForestClassifier	1.000000	0.000000	0.000000	0.000000
0	LinearRegression	NaN	16.245047	423.599554	20.581534

Binary Popularity Prediction

	Model	Accuracy	MAE	MSE	RMSE
1	RandomForestClassifier	0.899355	1.635222	0.101052	0.317887
0	LogisticRegression	0.489243	123.985612	0.510757	0.714672
4	AdaBoostClassifier	0.247302	0.097251	0.097251	0.311852
2	KNeighborsClassifier	0.097251	230.200882	0.902749	0.950131
3	DecisionTreeClassifier	0.097251	230.200882	0.902749	0.950131

Model Performance and Hyperparameters

Recall Curve for Binary Predictor



Model Performance and Hyperparameters

Popularity Score Prediction: Linear Regression

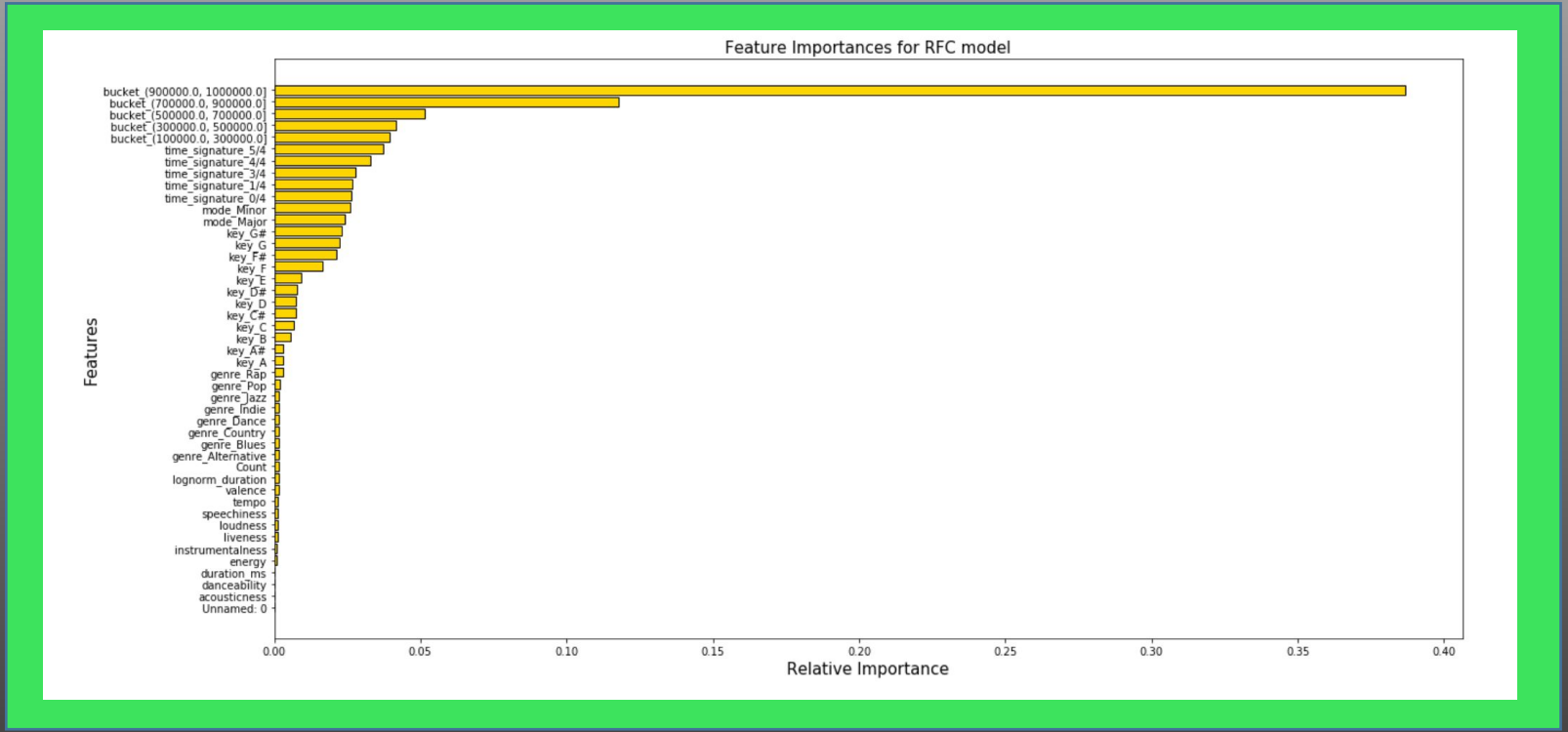
```
In [190]: LR_Model = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,normalize=False)
          LR_Model.fit(X_train, y_train)
          y_pred = LR_Model.predict(X_test)


          LR_MAE = metrics.mean_absolute_error(y_test, y_pred)
          LR_MSE = metrics.mean_squared_error(y_test, y_pred)
          LR_RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
          print('Mean Absolute Error:', LR_MAE)
          print('Mean Squared Error:', LR_MSE)
          print('Root Mean Squared Error:', LR_RMSE)

          cv_scores_rf_test= cv_scores_test.mean()
          cv_scores_rf_train= cv_scores_train.mean()

          print ('Mean cross validation test score: ' +str(cv_scores_rf_test))
          print ('Mean cross validation train score: ' +str(cv_scores_rf_train))

          Mean Absolute Error: 6.237420071514329
          Mean Squared Error: 71.77087073161633
          Root Mean Squared Error: 8.471769043807576
          Mean cross validation test score: 0.955077746550893
          Mean cross validation train score: 0.9644744855320806
```

Binary Popularity Prediction: Random Forest

```
In [32]: param_grid = {'bootstrap': [True],
           'max_depth': [10, 40],
           'max_features': [2, 3],
           'min_samples_leaf': [2, 4],
           'min_samples_split': [5, 10],
           'n_estimators': [600, 1000]}
         rf = RandomForestClassifier()
         rf_cv= GridSearchCV(rf,param_grid,cv=5)
         rf_cv.fit(X,y)

Out[32]: GridSearchCV(cv=5, error_score=nan,
                      estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                                       class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       max_features='auto',
                                                       max_leaf_nodes=None,
                                                       max_samples=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       n_estimators=100, n_jobs=None,
                                                       oob_score=False,
                                                       random_state=None, verbose=0,
                                                       warm_start=False),
                      iid='deprecated', n_jobs=None,
                      param_grid={'bootstrap': [True], 'max_depth': [10, 40],
                                  'max_features': [2, 3], 'min_samples_leaf': [2, 4],
                                  'min_samples_split': [5, 10],
                                  'n_estimators': [600, 1000]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

Best Parameters: learning_rate': 0.01, 'n_estimators': 800 and bootstrap': True, 'max_depth': 40, 'max_features': 2, 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 1000

Important Features



Key Takeaways

- All genres (except for Blues) the more danceable a song is the more likely it is to be popular
- Trying to predict popularity alone was best done any the Linear Regression model with MAE of 6.26 and RMSE of 8.65
- Trying to predict if a song will be popular was best predicted by Random Forest or not has a 90% chance of being correct and MAE of 1.64 and RMSE of 0.32
- Features that contributed most to the popularity of a song for the Random Forest model are the time buckets and can be seen on the previous slide

- Look into different features and/or feature reduction to see if these features
- Designation of popularity at 80 could be arbitrary and changing to a different number could yield better results
- Training and testing data on models that look exclusively at one genre could lead to less noise in the data and to better predictive power.

Questions?