Supervised Learning

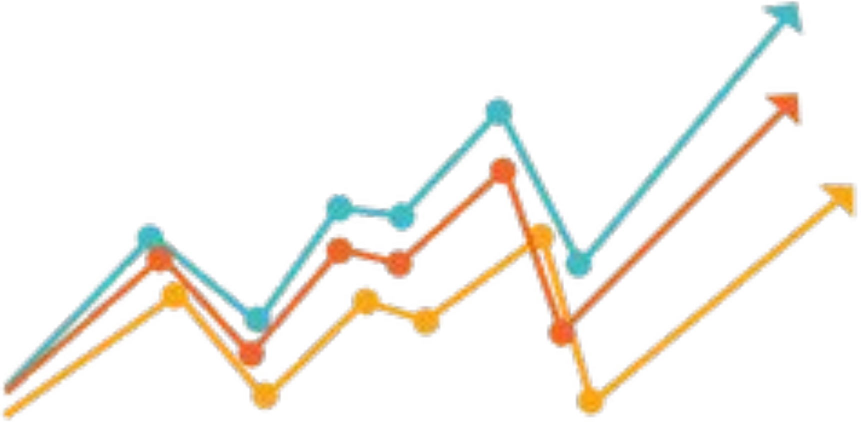
Final Project

Data: Most Streamed Spotify Songs 2023

Questions

What patterns exist among the top streamed songs?

Is there a way to predict if a song will be popular based on features on the song?



Data: EDA

Take a look at the data

	track_name	artist(s)_name	artist_count	released_year	released_month	released_day	in_spotify_playlists	in_spotify_charts	streams	in_apple_playlists	...
0	Seven (feat. Latto) (Explicit Ver.)	Latto, Jung Kook	2	2023	7	14	553	147	141381703	43	...
1	LALA	Myke Towers	1	2023	3	23	1474	48	133716286	48	...
2	vampire	Olivia Rodrigo	1	2023	6	30	1397	113	140003974	94	...
3	Cruel Summer	Taylor Swift	1	2019	8	23	7858	100	800840817	116	...
4	WHERE SHE GOES	Bad Bunny	1	2023	5	18	3133	50	303236322	84	...

Data: EDA

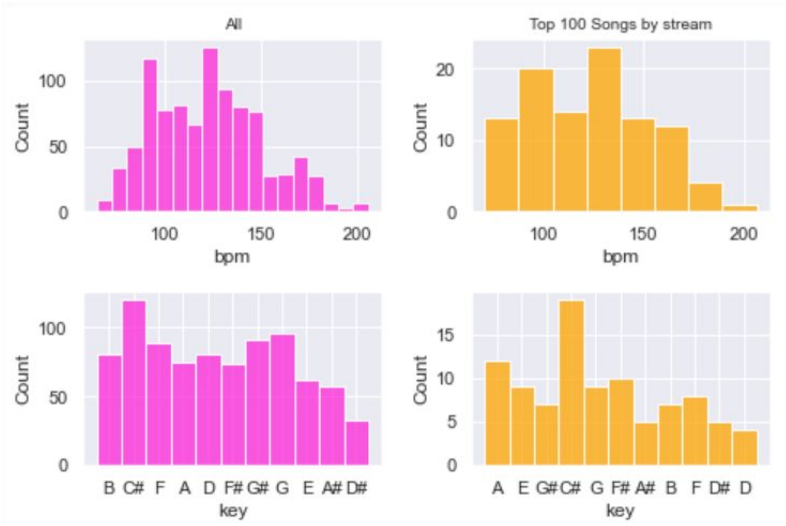
Take a look at the data types

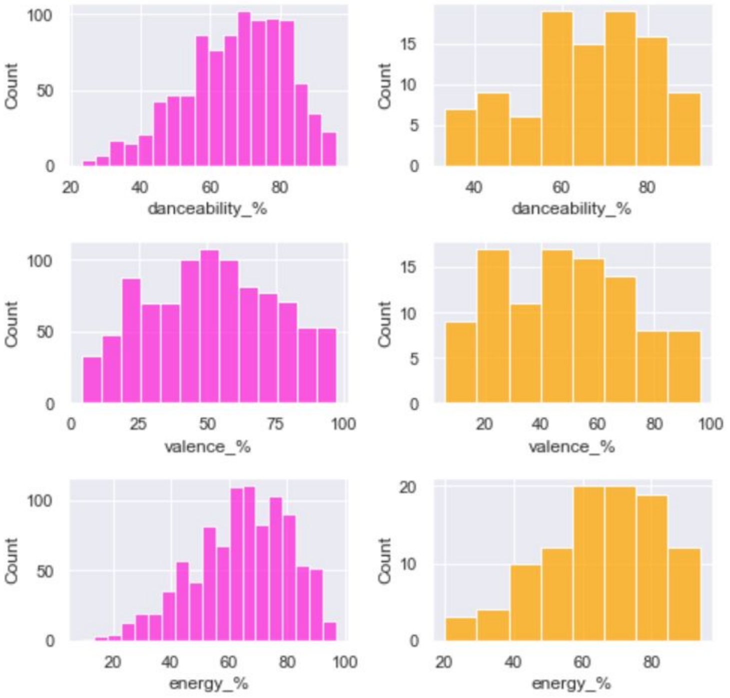
Determine if there are null values

```
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               953 non-null    object
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
 17  danceability_%        953 non-null    int64
 18  valence_%             953 non-null    int64
...
 23  speechiness_%         953 non-null    int64
dtypes: int64(17), object(7)
```

Data: EDA

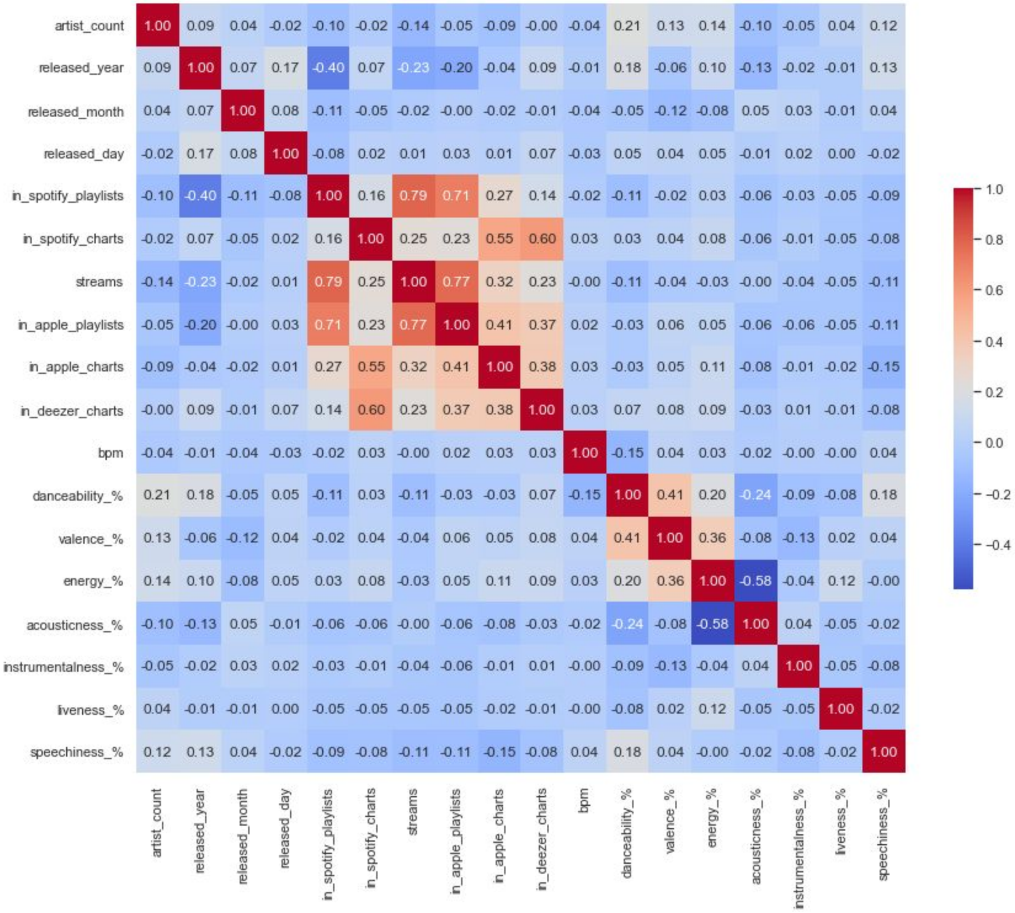
Compare values for all songs vs top 100 songs





Data: EDA

Take a look at the correlation matrix



Linear Regression

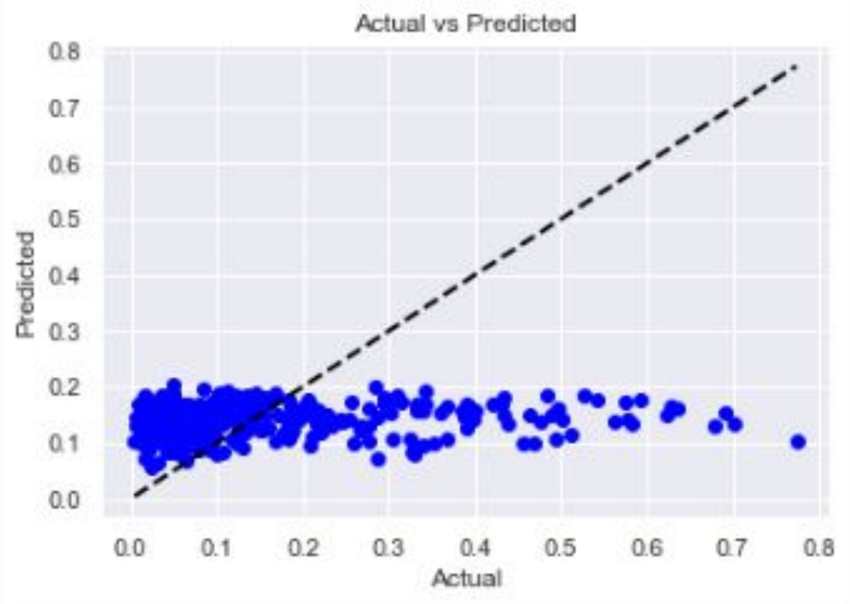
```
y = data['streams']
X = data[['bpm',
 'key_A', 'key_A#', 'key_B',
        'key_C#', 'key_D', 'key_D#', 'key_E', 'key_F', 'key_F#', 'key_G',
        'key_G#', 'mode_Major', 'mode_Minor',
 'danceability_%',
 'valence_%',
 'energy_%',
 'acousticness_%',
 'instrumentalness_%',
 'liveness_%',
 'speechiness_%' ]]

# Normalize the target
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
y_normalized = scaler.fit_transform(y.values.reshape(-1, 1))

X_train, X_test, y_train, y_test = train_test_split(X, y_normalized, test_size=0.4, random_state=20)

model = LinearRegression()
model.fit(X_train,y_train)

predictions = model.predict(X_test)
print(
  'mean_squared_error : ', mean_squared_error(y_test, predictions))
print(
  'mean_absolute_error : ', mean_absolute_error(y_test, predictions))
print(
  'r2_score :', r2_score(y_test, predictions))
```

Target variable is ‘streams’

Normalize ‘streams’ using MinMax

Linear Regression

```
mean_squared_error :  0.022256678484159128
mean_absolute_error :  0.11116957026618782
r2_score : 0.021046219353758833
```



Linear regression does not work well here at all

This is not surprising

The features are not linearly related

Trees

Decision Tree & Random Forest

Using sklearn

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train[['danceability_%',
 'valence_%',
 'energy_%',
 'acousticness_%',
 'instrumentalness_%',
 'liveness_%',
 'speechiness_%']] = scaler.fit_transform(X_train[['danceability_
 'valence_%',
 'energy_%',
 'acousticness_%',
 'instrumentalness_%',
 'liveness_%',
 'speechiness_%']])

X_test[['danceability_%',
 'valence_%',
 'energy_%',
 'acousticness_%',
 'instrumentalness_%',
 'liveness_%',
 'speechiness_%']] = scaler.transform(X_test[['danceability_%',
 'valence_%',
 'energy_%',
 'acousticness_%',
 'instrumentalness_%',
 'liveness_%',
 'speechiness_%']])
```

```
tree_model = DecisionTreeRegressor()
rf_model = RandomForestRegressor()

tree_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)


tree_mse = mean_squared_error(y_train, tree_model.predict(X_train))
tree_mae = mean_absolute_error(y_train, tree_model.predict(X_train))
rf_mse = mean_squared_error(y_train, rf_model.predict(X_train))
rf_mae = mean_absolute_error(y_train, rf_model.predict(X_train))

RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)


from math import sqrt

print("Decision Tree training mse = ",tree_mse," & mae = ",tree_mae," & rmse = ", sqrt(tree_mse))
print("Random Forest training mse = ",rf_mse," & mae = ",rf_mae," & rmse = ", sqrt(rf_mse))
```

```
Decision Tree training mse =  0.0  & mae =  0.0  & rmse =  0.0
Random Forest training mse =  0.0034531586946030925  & mae =  0.04291456664272159  & rmse =  0.058763583064710176
```

Trees

Results

```
tree_test_mse = mean_squared_error(y_test, tree_model.predict(X_test))
tree_test_mae = mean_absolute_error(y_test, tree_model.predict(X_test))
rf_test_mse = mean_squared_error(y_test, rf_model.predict(X_test))
rf_test_mae = mean_absolute_error(y_test, rf_model.predict(X_test))

print("Decision Tree test mse = ",tree_test_mse," & mae = ",tree_test_mae," & rmse = ", sqrt(tree_test_mse))
print("Random Forest test mse = ",rf_test_mse," & mae = ",rf_test_mae," & rmse = ", sqrt(rf_test_mse))
```

4]

```
Decision Tree test mse =  0.049148873249189186  & mae =  0.1494445308737565  & rmse =  0.22169545157532933
Random Forest test mse =  0.023846587031924858  & mae =  0.11659426355051591  & rmse =  0.15442340182732944
```

Conclusions & Next Steps

Decision Tree overfits the training data

Random Forest improves upon this

Complex patterns & relationships among features

More feature engineering can be performed

Ensemble & Deep learning techniques could be applied