Individual Project

Danel Tukibay

School of Sciences and Humanities, Nazarbayev University

MATH 446: Time Series Analysis

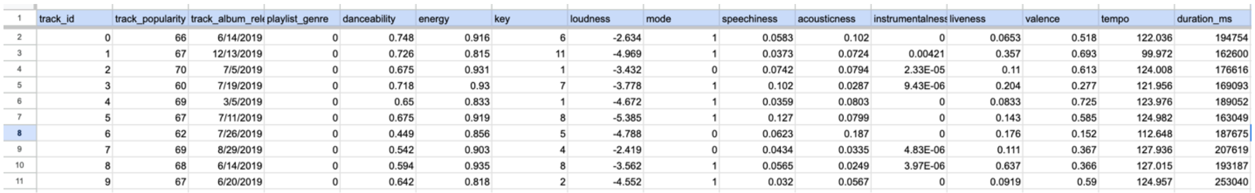
Kerem Ugurlu

November 30, 2023

Applying LDA and PCA to Playlist Genre Classification

Introduction

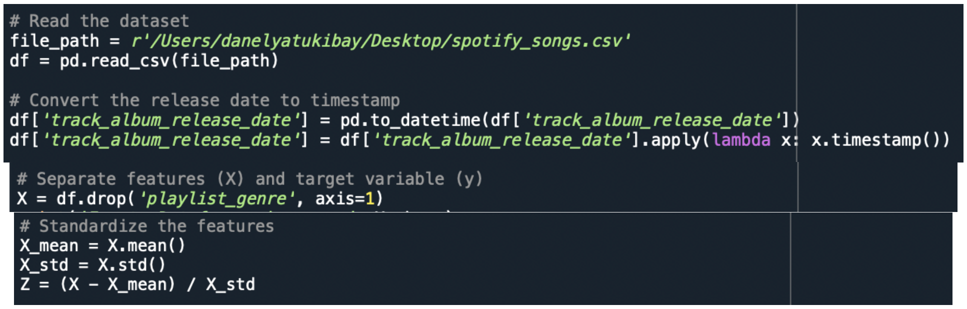
In this project, I applied Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) to a dataset(https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs) containing information about 30,000 songs, including various features like track popularity, danceability, energy, and others. Irrelevant features such as track name, artist name, album name were removed as they contain non-numerical information. The goal was to classify songs into different playlist genres such as pop, hip-hop, rap and etc based on the features mentioned above.



1	track_id	track_popularity	track_album_rel	playlist_genre	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
2	0	66	6/14/2019	0	0.748	0.916	6	-2.634	1	0.0583	0.102	0	0.0653	0.518	122.036	194754
3	1	67	12/13/2019	0	0.726	0.815	11	-4.969	1	0.0373	0.0724	0.00421	0.357	0.693	99.972	162600
4	2	70	7/5/2019	0	0.675	0.931	1	-3.432	0	0.0742	0.0794	2.33E-05	0.11	0.613	124.008	176616
5	3	60	7/19/2019	0	0.718	0.93	7	-3.778	1	0.102	0.0287	9.43E-06	0.204	0.277	121.956	169093
6	4	69	3/5/2019	0	0.65	0.833	1	-4.672	1	0.0359	0.0803	0	0.0833	0.725	123.976	189052
7	5	67	7/11/2019	0	0.675	0.919	8	-5.385	1	0.127	0.0799	0	0.143	0.585	124.982	163049
8	6	62	7/26/2019	0	0.449	0.856	5	-4.788	0	0.0623	0.187	0	0.176	0.152	112.648	187675
9	7	69	8/29/2019	0	0.542	0.903	4	-2.419	0	0.0434	0.0335	4.83E-06	0.111	0.367	127.936	207619
10	8	68	6/14/2019	0	0.594	0.935	8	-3.562	1	0.0565	0.0249	3.97E-06	0.637	0.366	127.015	193187
11	9	67	6/20/2019	0	0.642	0.818	2	-4.552	1	0.032	0.0567	0	0.0919	0.59	124.957	253040

Data Preprocessing

I loaded the dataset and performed some initial data preprocessing steps, such as converting the release date to a numerical format, and standardizing the features to have a mean of 0 and a standard deviation of 1.

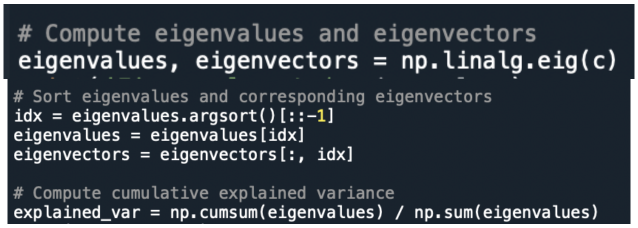


```
# Read the dataset
file_path = r'/Users/danelyatukibay/Desktop/spotify_songs.csv'
df = pd.read_csv(file_path)

# Convert the release date to timestamp
df['track_album_release_date'] = pd.to_datetime(df['track_album_release_date'])
df['track_album_release_date'] = df['track_album_release_date'].apply(lambda x: x.timestamp())

# Separate features (X) and target variable (y)
X = df.drop('playlist_genre', axis=1)
# Standardize the features
X_mean = X.mean()
X_std = X.std()
Z = (X - X_mean) / X_std
```

Principal Component Analysis (PCA)

Covariance Matrix and Eigenvalues:

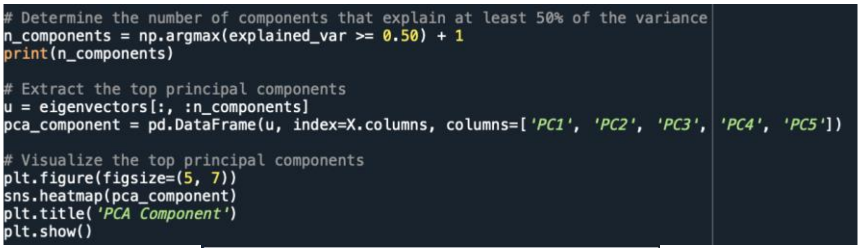
I computed the covariance matrix and eigenvalues of the standardized feature matrix. The eigenvalues and eigenvectors were used to determine the principal components and their cumulative explained variance.



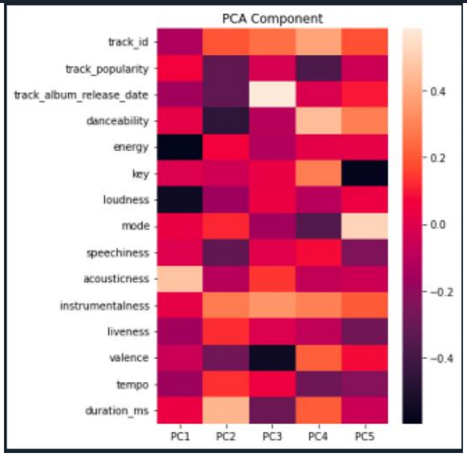
```
# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(c)
# Sort eigenvalues and corresponding eigenvectors
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]

# Compute cumulative explained variance
explained_var = np.cumsum(eigenvalues) / np.sum(eigenvalues)
```

Dimensionality Reduction:

I selected the number of principal components that explained at least 50% of the variance in the data. Using these components, I visualized their importance using a heatmap.

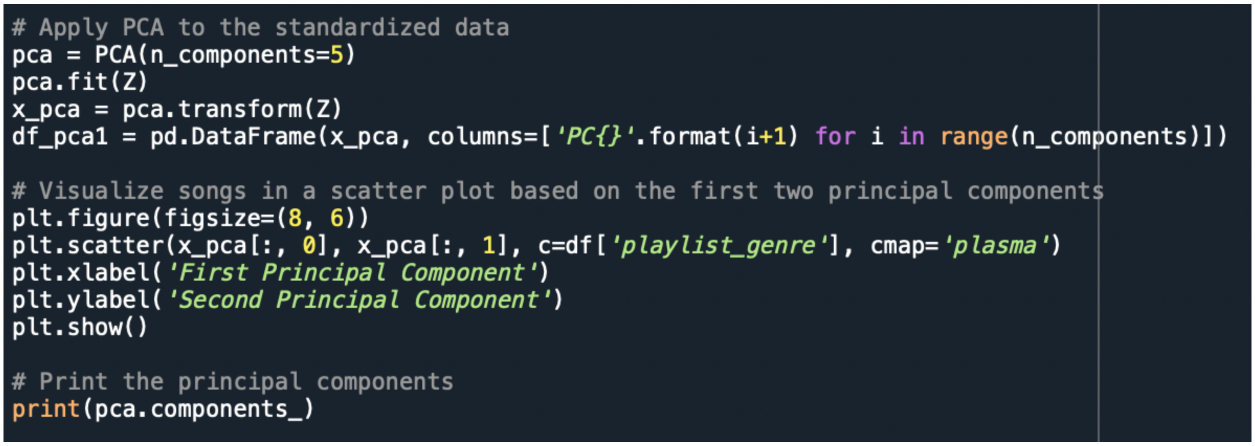


```
# Determine the number of components that explain at least 50% of the variance
n_components = np.argmax(explained_var >= 0.50) + 1
print(n_components)

# Extract the top principal components
u = eigenvectors[:, :n_components]
pca_component = pd.DataFrame(u, index=X.columns, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])

# Visualize the top principal components
plt.figure(figsize=(5, 7))
sns.heatmap(pca_component)
plt.title('PCA Component')
plt.show()
```

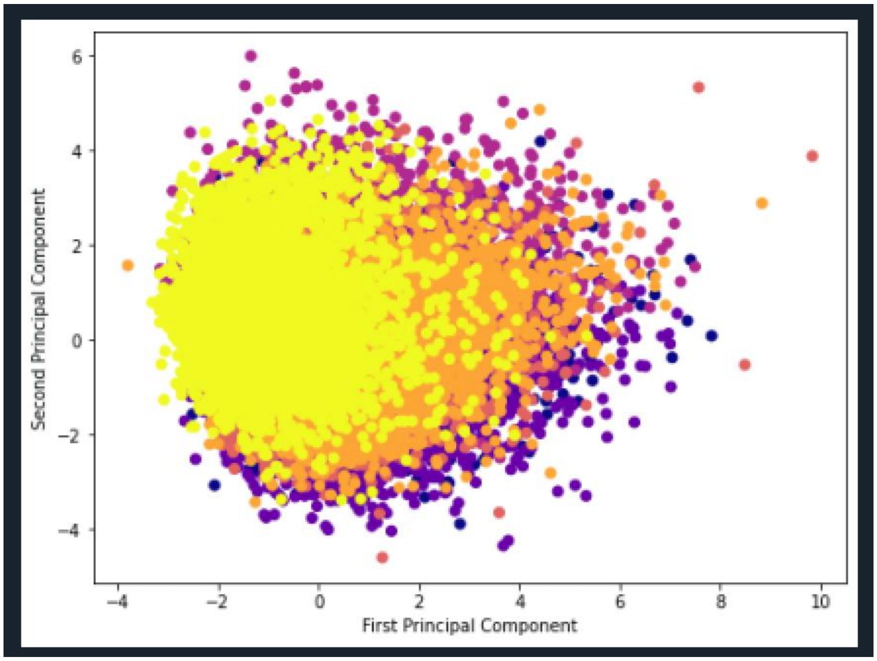


Visualization:

I applied PCA to the data, reducing it to five principal components, and visualized the songs in a scatter plot based on the first two principal components. The colors represent different playlist genres.



```python
# Apply PCA to the standardized data
pca = PCA(n_components=5)
pca.fit(Z)
x_pca = pca.transform(Z)
df_pca1 = pd.DataFrame(x_pca, columns=['PC{}'.format(i+1) for i in range(n_components)])

# Visualize songs in a scatter plot based on the first two principal components
plt.figure(figsize=(8, 6))
plt.scatter(x_pca[:, 0], x_pca[:, 1], c=df['playlist_genre'], cmap='plasma')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.show()

# Print the principal components
print(pca.components_)
```

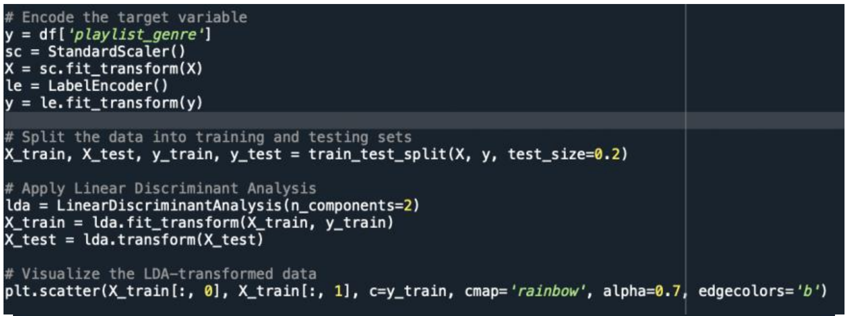


Linear Discriminant Analysis (LDA)

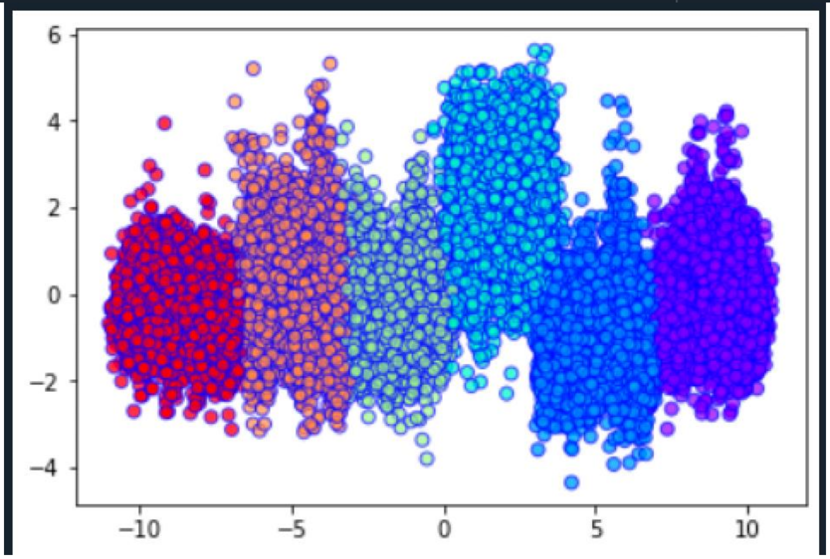
I then applied Linear Discriminant Analysis to the standardized feature matrix.

Classification

I used LDA to transform the features into a lower-dimensional space and visualized the songs in a scatter plot based on the first two discriminant functions.

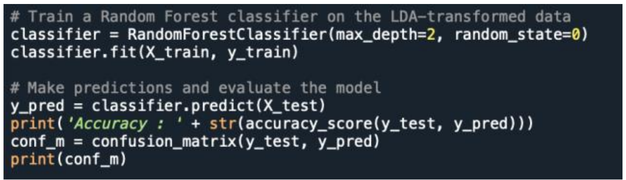


```
# Encode the target variable
y = df['playlist_genre']
sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Apply Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

# Visualize the LDA-transformed data
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='rainbow', alpha=0.7, edgecolors='b')
```

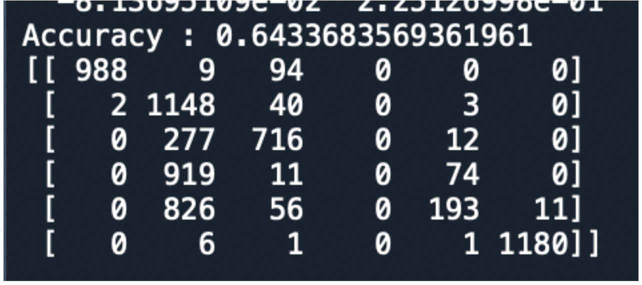


Random Forest Classification:

To further assess the effectiveness of LDA, I employed a Random Forest classifier on the LDA-transformed data and evaluated its accuracy and confusion matrix.



```
# Train a Random Forest classifier on the LDA-transformed data
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Make predictions and evaluate the model
y_pred = classifier.predict(X_test)
print('Accuracy : ' + str(accuracy_score(y_test, y_pred)))
conf_m = confusion_matrix(y_test, y_pred)
print(conf_m)
```



```
 -8.13693109e-02  2.25126998e-01
Accuracy : 0.6433683569361961
[[ 988    9   94    0    0    0]
 [    2 1148   40    0    3    0]
 [    0  277  716    0   12    0]
 [    0  919   11    0   74    0]
 [    0  826   56    0  193   11]
 [    0    6    1    0    1 1180]]
```

Results and Comparison

The accuracy of the Random Forest classifier applied on the LDA-transformed data was approximately 64.34%. This provides a baseline for comparison with the results obtained using PCA.

Conclusion

In conclusion, both PCA and LDA were applied to the dataset for dimensionality reduction and classification. The choice between PCA and LDA depends on the specific goals of the analysis. PCA captures overall variance, while LDA aims to maximize class separation. The results indicate that LDA, in combination with a Random Forest classifier, achieved a 64.34% accuracy in predicting playlist genres.