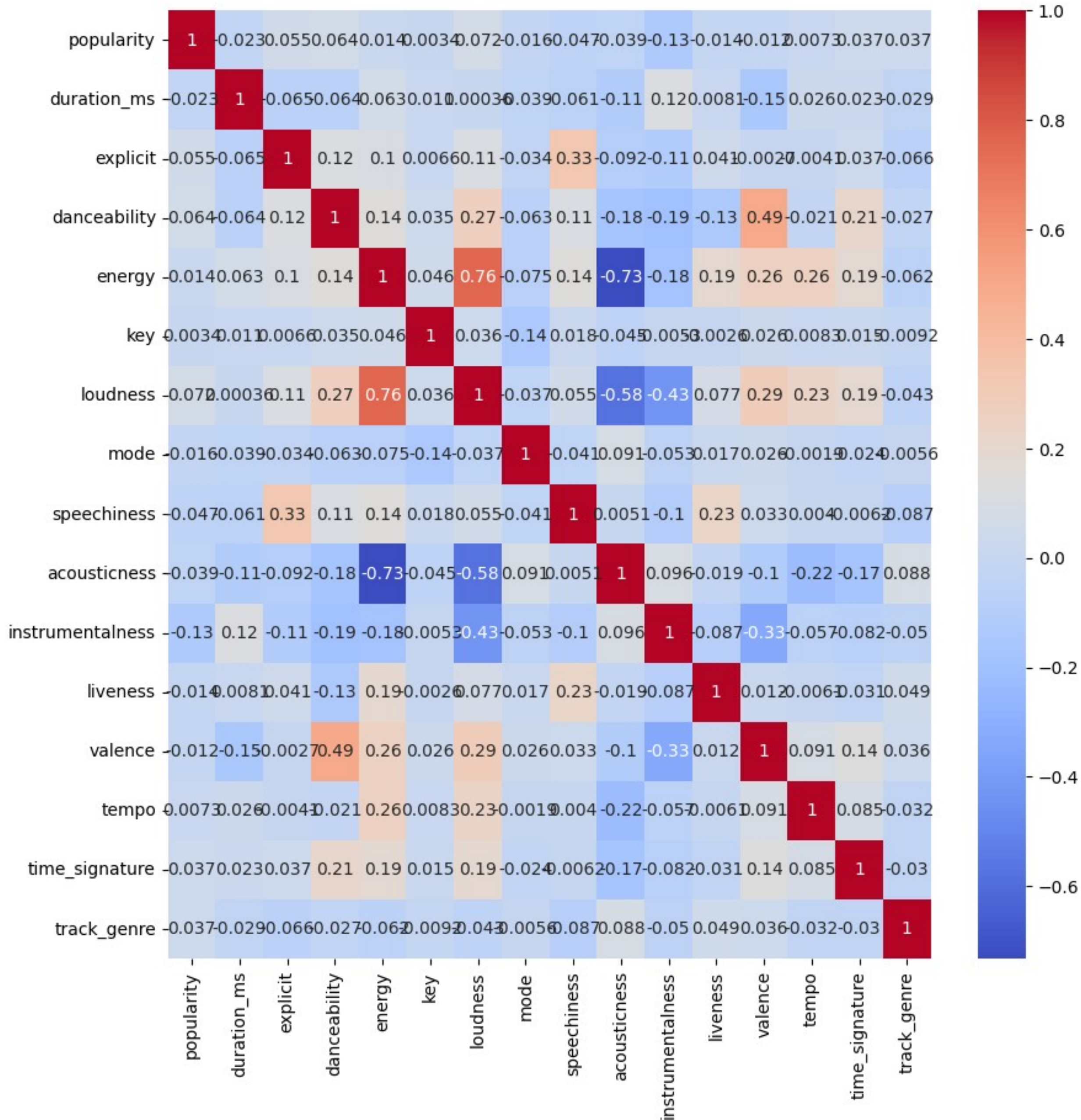
Project Check-in 2

```
%pip install scikit-lego
%pip install seaborn
#%pip install nbstripout
#%nbstripout --install
```

```
Requirement already satisfied: scikit-lego in c:\users\isaac\appdata\
local\programs\python\python311\lib\site-packages (0.9.1)
Requirement already satisfied: narwhals>=1.0.0 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
scikit-lego) (1.9.3)
Requirement already satisfied: pandas>=1.1.5 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
scikit-lego) (2.1.2)
Requirement already satisfied: scikit-learn>=1.0 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
scikit-lego) (1.3.2)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
pandas>=1.1.5->scikit-lego) (1.26.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
isaac\appdata\roaming\python\python311\site-packages (from
pandas>=1.1.5->scikit-lego) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\isaac\appdata\
local\programs\python\python311\lib\site-packages (from pandas>=1.1.5-
>scikit-lego) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
pandas>=1.1.5->scikit-lego) (2023.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\isaac\appdata\
local\programs\python\python311\lib\site-packages (from scikit-
learn>=1.0->scikit-lego) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
scikit-learn>=1.0->scikit-lego) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\isaac\
appdata\local\programs\python\python311\lib\site-packages (from
scikit-learn>=1.0->scikit-lego) (3.2.0)
Requirement already satisfied: six>=1.5 in c:\users\isaac\appdata\
roaming\python\python311\site-packages (from python-dateutil>=2.8.2-
>pandas>=1.1.5->scikit-lego) (1.16.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip available: 22.3 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: seaborn in c:\users\isaac\appdata\
local\programs\python\python311\lib\site-packages (0.13.2)
```

```
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (1.26.1)
Requirement already satisfied: pandas>=1.2 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (2.1.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from seaborn) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\isaac\appdata\roaming\python\python311\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\isaac\appdata\roaming\python\python311\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\isaac\appdata\local\programs\python\python311\lib\site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\isaac\appdata\roaming\python\python311\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip available: 22.3 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip

import numpy as np
import matplotlib.pyplot as plt
```

```python
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler,
PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error


df = pd.read_csv("./dataset.csv")

# Drop all duplicates (need to remove the first column because these
are just indices)
revised_df = df.drop(columns='Unnamed:
0').drop_duplicates(subset=['track_id','album_name','artists','track_n
ame'])

# Drop NaN values
# NEED TO FIGURE OUT WHETHER THIS IS WORTH IT BC ONLY
ARTISTS/ALBUM_NAME/TRACK_NAME ARE NAN

revised_df.dropna(axis=0,inplace=True)

revised_df.drop(columns=['track_id', 'artists', 'album_name',
'track_name'], inplace=True)
columns = revised_df.columns
le = LabelEncoder()
revised_df['track_genre'] =
le.fit_transform(revised_df['track_genre'])

scaler = StandardScaler()
revised_df = scaler.fit_transform(revised_df)
revised_df = pd.DataFrame(revised_df, columns=columns)

# Look at linear correlations between features
corr_matrix = revised_df.corr(method='pearson')
plt.figure(figsize=(10 , 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```




```
np.random.seed(42)
x_train, x_test, y_train, y_test =
train_test_split(revised_df['loudness'], revised_df['energy'],
test_size=0.2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

(71792,)
(71792,)
```

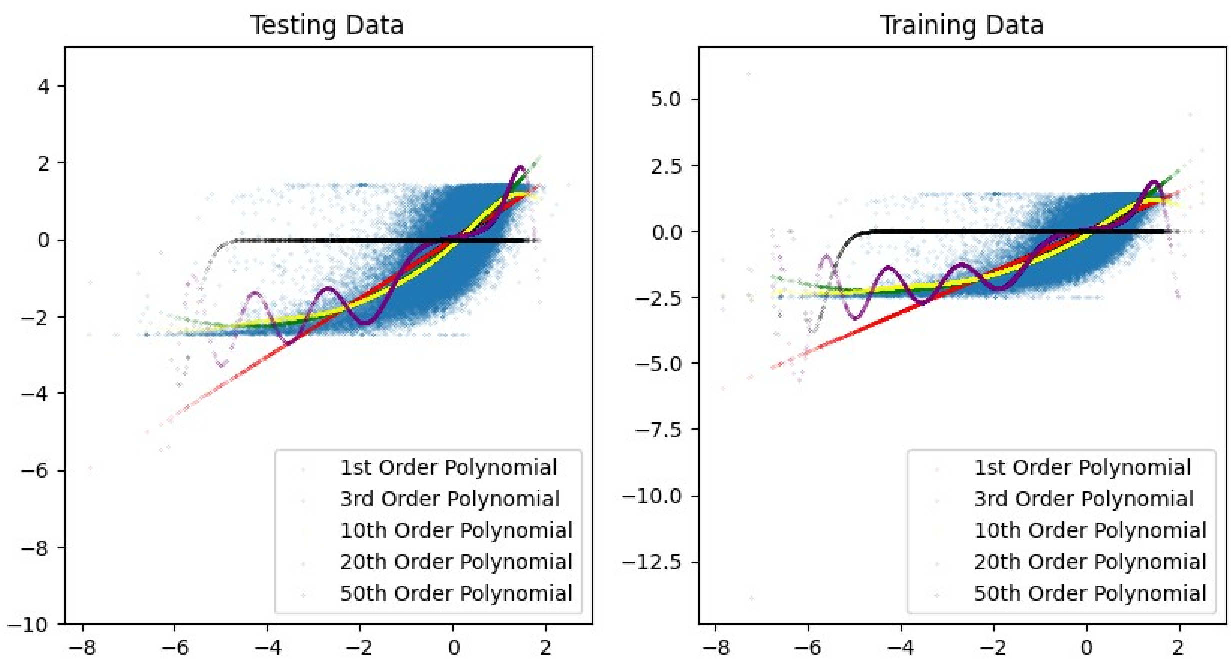
```
(17948,)
(17948,)
```

1. and 2. : We are using loudness as a predictor of energy because they have a high linear correlation coefficient, as seen in the above graphic.

```
np.random.seed(1000)

x_train, x_test, y_train, y_test =
train_test_split(revised_df['loudness'], revised_df['energy'],
test_size=0.2)

polynomial_orders = [(1, "1st"), (3, "3rd"), (10, "10th"), (20,
"20th"), (50, "50th")]
colors = ['red', 'green', 'yellow', 'purple', 'black']
fig, ax = plt.subplots(1,2,figsize=(10,5))
ax[0].scatter(x_train, y_train, s=0.01)
ax[1].scatter(x_train, y_train, s=0.01)

x_train = x_train.to_numpy()
x_test = x_test.to_numpy()
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()


sorted_train_indices = np.argsort(x_train)
sorted_test_indices = np.argsort(x_test)
x_train = x_train[sorted_train_indices]
x_test = x_test[sorted_test_indices]
y_train = y_train[sorted_train_indices]
y_test = y_test[sorted_test_indices]

poly_reg_model = LinearRegression()

regression_data = []

for ind,i in enumerate(polynomial_orders):
    poly = PolynomialFeatures(degree=i[0], include_bias=False)
    poly_features_train = poly.fit_transform(x_train.reshape(-1,1))
    poly_features_test = poly.fit_transform(x_test.reshape(-1,1))
    poly_reg_model.fit(poly_features_train, y_train)

    energy_predicted_test = poly_reg_model.predict(poly_features_test)
    energy_predicted_train =
poly_reg_model.predict(poly_features_train)

    ax[0].scatter(x_test, energy_predicted_test, c=colors[ind],
label=f"{i[1]} Order Polynomial", s=0.01)
    ax[1].scatter(x_train, energy_predicted_train, c=colors[ind],
label=f"{i[1]} Order Polynomial", s=0.01)
```

```
    regression_data.append([i,energy_predicted_train,
energy_predicted_test])

# Both graphs have a scatterplot of the training data underneath the
regression curves
ax[0].legend()
ax[1].legend()
ax[0].set_title("Testing Data")
ax[1].set_title("Training Data")
ax[0].set_ylim(-10,5)
```

```
(-10.0, 5.0)
```



1. Here we see evidence of underfitting for the 1st, 3rd, and to a lesser degree, 10th order polynomial regressions. This is shownby the fact that those polynomials have high error on even the training data. The 20th and 50th order show signs of overfitting, namely, high-frequency components present when the underlying trend is clearly dominated by low-frequency signals.

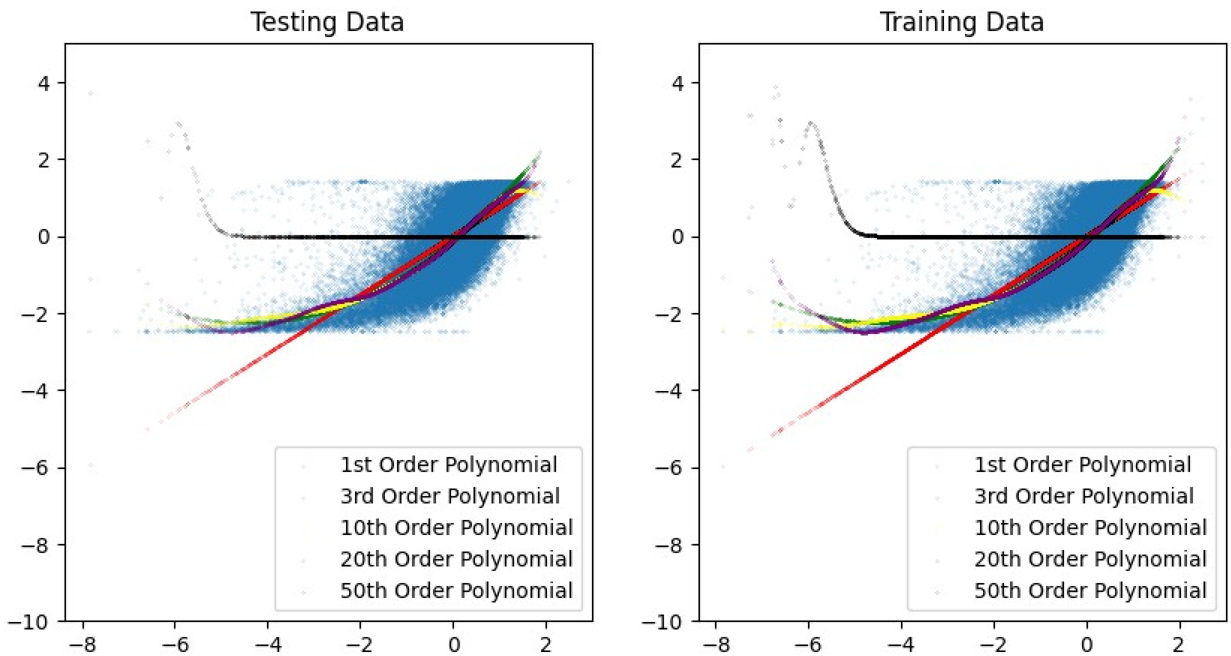
```
for data in regression_data:
    rmse_train = np.sqrt(mean_squared_error(y_train, data[1]))
    rmse_test = np.sqrt(mean_squared_error(y_test, data[2]))
    print(f"{data[0][1]} Order Polynomial:")
    print(f"    RMSE for Training Data: {rmse_train}")
    print(f"    RMSE for Testing Data: {rmse_test}")
```

```
1st Order Polynomial:
    RMSE for Training Data: 0.6515607320666389
    RMSE for Testing Data: 0.6505317330994379
```

```
3rd Order Polynomial:
    RMSE for Training Data: 0.6150049906076769
    RMSE for Testing Data: 0.6156649809737508
10th Order Polynomial:
    RMSE for Training Data: 0.6121803280236077
    RMSE for Testing Data: 0.6125447572079118
20th Order Polynomial:
    RMSE for Training Data: 0.7006015814540063
    RMSE for Testing Data: 2.3490416750976397
50th Order Polynomial:
    RMSE for Training Data: 0.9968783464993808
    RMSE for Testing Data: 220.4679486241852
```

```
ridge = Ridge()

polynomial_orders = [(1, "1st"), (3, "3rd"), (10, "10th"), (20,
"20th"), (50, "50th")]
colors = ['red', 'green', 'yellow', 'purple', 'black']
fig, ax = plt.subplots(1,2,figsize=(10,5))
ax[0].scatter(x_train, y_train, s=0.01)
ax[1].scatter(x_train, y_train, s=0.01)

ridge_regression_data = []

for ind,i in enumerate(polynomial_orders):
    poly = PolynomialFeatures(degree=i[0], include_bias=False)
    poly_features_train = poly.fit_transform(x_train.reshape(-1,1))
    poly_features_test = poly.fit_transform(x_test.reshape(-1,1))
    ridge.fit(poly_features_train, y_train)

    energy_predicted_test = ridge.predict(poly_features_test)
    energy_predicted_train = ridge.predict(poly_features_train)

    ax[0].scatter(x_test, energy_predicted_test, c=colors[ind],
label=f"{i[1]} Order Polynomial", s=0.01)
    ax[1].scatter(x_train, energy_predicted_train, c=colors[ind],
label=f"{i[1]} Order Polynomial", s=0.01)
    ridge_regression_data.append([i,energy_predicted_train,
energy_predicted_test])

# Both graphs have a scatterplot of the training data underneath the
regression curves
ax[0].legend()
ax[1].legend()
ax[0].set_title("Testing Data")
ax[1].set_title("Training Data")
ax[0].set_ylim(-10,5)
ax[1].set_ylim(-10,5)
```

```
(-10.0, 5.0)
```




```
for data in ridge_regression_data:
    rmse_train = np.sqrt(mean_squared_error(y_train, data[1]))
    rmse_test = np.sqrt(mean_squared_error(y_test, data[2]))
    print(f"{data[0][1]} Order Polynomial:")
    print(f"    RMSE for Training Data: {rmse_train}")
    print(f"    RMSE for Testing Data: {rmse_test}")

1st Order Polynomial:
    RMSE for Training Data: 0.6515607321524421
    RMSE for Testing Data: 0.6505316607044392
3rd Order Polynomial:
    RMSE for Training Data: 0.6150049909485981
    RMSE for Testing Data: 0.6156647508729066
10th Order Polynomial:
    RMSE for Training Data: 0.6121803307371144
    RMSE for Testing Data: 0.6125449484876924
20th Order Polynomial:
    RMSE for Training Data: 0.6168082363245457
    RMSE for Testing Data: 0.6153165937239949
50th Order Polynomial:
    RMSE for Training Data: 1.0150927233999276
    RMSE for Testing Data: 114.86896962632257
```

1. With L2 regularization, the model does worse on training data and better on testing data than the regression without regularization, as expected, because overfitting is reduced.