Music_popularity

May 14, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

[13]:
```
import sys

if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

[8]:
```
dataset = pd.read_csv('Downloads/data_spotify.csv')
```

[9]:
```
dataset.head()
```

[9]:

	acousticness	danceability	duration_ms	energy	instrumentalness	key
0	0.982	0.279	831667	0.211	0.878000	10
1	0.732	0.819	180533	0.341	0.000000	7
2	0.961	0.328	500062	0.166	0.913000	3
3	0.967	0.275	210000	0.309	0.000028	5
4	0.957	0.418	166693	0.193	0.000002	3

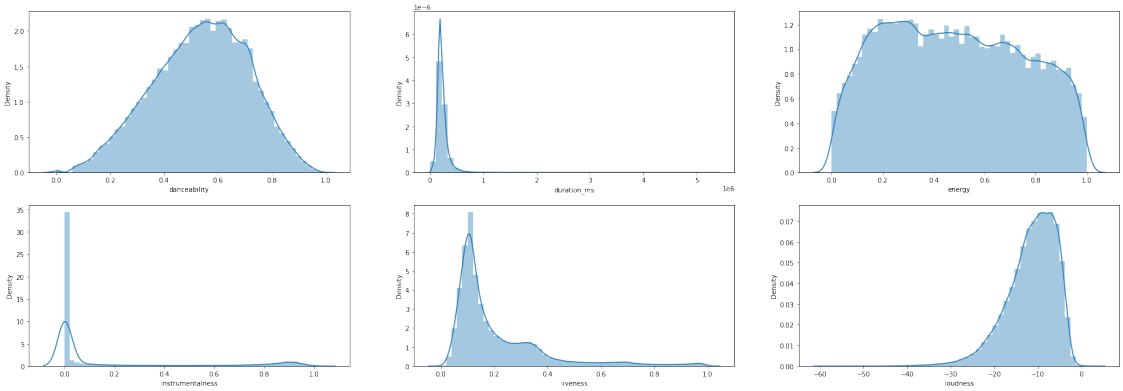
	liveness	loudness	mode	popularity	release_date	speechiness	tempo
0	0.665	-20.096	1	4	1921	0.0366	80.954
1	0.160	-12.441	1	5	1921	0.4150	60.936
2	0.101	-14.850	1	5	1921	0.0339	110.339
3	0.381	-9.316	1	3	1921	0.0354	100.109
4	0.229	-10.096	1	2	1921	0.0380	101.665

	valence	explicit	year	artists
0	0.0594	0	1921	['Sergei Rachmaninoff', 'James Levine', 'Berli…
1	0.9630	0	1921	['Dennis Day']
2	0.0394	0	1921	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi…
3	0.1650	0	1921	['Frank Parker']
4	0.2530	0	1921	['Phil Regan']

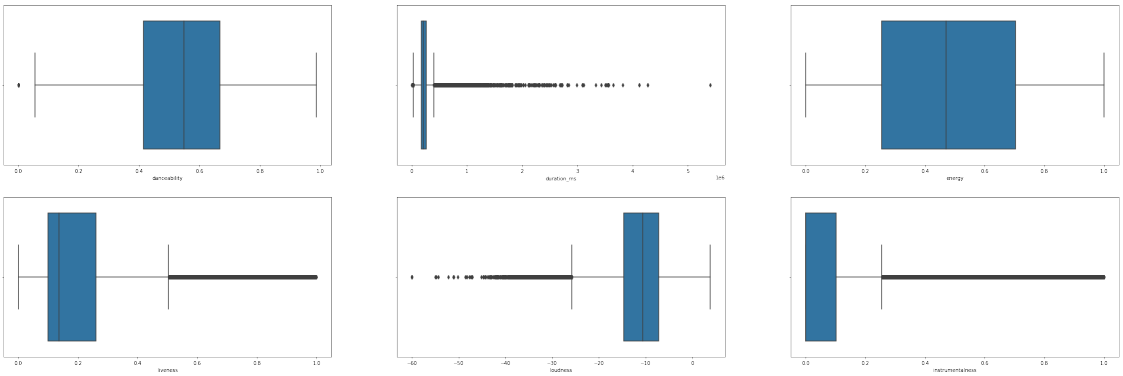
[11]:
```
dataset.shape
```

[11]:
```
(170653, 17)
```

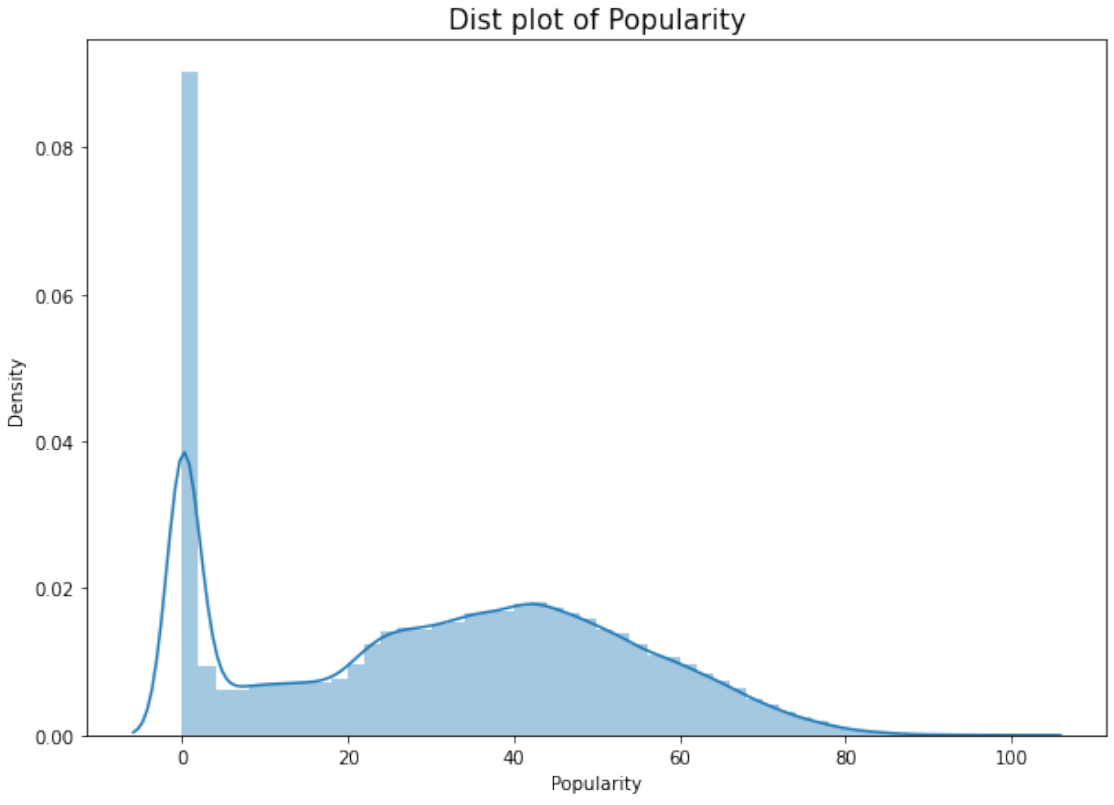
[14]:
```
plt.figure(figsize = (30, 10))
plt.subplot(231)
sns.distplot(dataset['danceability'])
plt.subplot(232)
sns.distplot(dataset['duration_ms'])
plt.subplot(233)
sns.distplot(dataset['energy'])
plt.subplot(234)
sns.distplot(dataset['instrumentalness'])
plt.subplot(235)
sns.distplot(dataset['liveness'])
plt.subplot(236)
sns.distplot(dataset['loudness'])
plt.show()
```



[15]:
```
plt.figure(figsize = (40, 20))
plt.subplot(331)
sns.boxplot(dataset['danceability'])
plt.subplot(332)
sns.boxplot(dataset['duration_ms'])
plt.subplot(333)
sns.boxplot(dataset['energy'])
plt.subplot(334)
sns.boxplot(dataset['liveness'])
plt.subplot(335)
sns.boxplot(dataset['loudness'])
plt.subplot(336)
sns.boxplot(dataset['instrumentalness'])
plt.show()
```



[16]:
```
plt.figure(figsize = (10, 7))
sns.distplot(dataset.popularity)
plt.title("Dist plot of Popularity", fontdict = {'fontsize' : 15})
plt.xlabel('Popularity')
plt.show()
plt.show()
```



[18]:
```
import plotly.express as px
def year(df):
    if df>1920 and df<=1945:
        return "Post-Great War"
    if df>1945 and df<=1970:
        return "Retro"
    if df>1970 and df<=1995:
        return "Modern"
    else:
        return "Post-Modern"
dataset['era'] = dataset['year'].apply(year)
px.pie(data_frame = dataset, names = 'era', hole = 0.2, title = 'Eras of Music')
```

[19]:
```
def func(df):
    if df == 1:
        return 'Yes'
    else:
        return 'No'
dataset['isExplicit'] = dataset['explicit'].apply(func)
px.pie(data_frame = dataset, names = 'isExplicit', hole = 0.2, title =␣
↪'Explicit')
```

[20]:
```
def loud(row):
    m=dataset['loudness'].median()
    sd=dataset['loudness'].std()
    if row['loudness']>=m+(1.5*sd):
        return "Extreme"
    elif row['loudness']>=m+(sd):
        return "Very Loud"
    elif row['loudness']>=m+(0.5*sd):
        return "Loud"
    elif row['loudness']>=m-(0.5*sd):
        return "Soft"
    elif row['loudness']>=m-(sd):
        return "Very Soft"
    else:
        return "Mellow"
dataset['is_loud']=dataset.apply(lambda row: loud(row), axis=1)
px.pie(data_frame = dataset, names = 'is_loud', hole = 0.2, title = 'IS LOUD')
```

[21]:
```
def energy(row):
    if row['energy']>=dataset['energy'].mean():
        return "High"
    else:
        return "Low"
dataset['en_type']=dataset.apply(lambda row: energy(row),axis=1)
px.pie(names=dataset['en_type'],hole=0.2)
```

[22]:
```
def func(df):
    if df > 75:
        return 'Very Popular'
    elif df > 50 and df < 76:
        return 'Popular'
    elif df > 25 and df < 51:
        return 'Average'
    else:
        return 'Not popular'
dataset['isPopular'] = dataset['popularity'].apply(func)
px.pie(data_frame = dataset, names = 'isPopular', hole = 0.2, title =␣
↪'Popularity')
```

[23]:
```
fig = px.scatter(dataset, x="loudness", y="danceability", size="liveness",
                 color="isExplicit", log_x=True,size_max=30)
fig.show()
```

[24]:
```
fig = px.scatter(dataset, x="loudness", y="danceability", size="popularity",
                 color="isExplicit", log_x=True,size_max=30)
fig.show()
```

[25]:
```
art=dataset
n=[]
g=[]
for name, group in art.groupby(['artists'])['popularity']:
    n.append(name)
    g.append(group.mean())
artist_pop=pd.DataFrame(n,g)
```

[26]:
```
artist_pop.columns=['Name']
```

[27]:
```
artist_pop['popularity']=artist_pop.index
artist_pop.sort_values(by='popularity',ascending=False,inplace=True)
px.bar(x=artist_pop['Name'].head(10),y=artist_pop['popularity'].head(10)).
↪update_layout(yaxis_title_text='Popularity')
```

[29]:
```
fig = px.scatter(dataset, x = "popularity", y = "danceability", ani␣
↪mation_frame = dataset['year'].sort_values(), animation_group = "isPopular",
           size = "key", color = "isExplicit", hover_name = "isPopular",
           log_x=True, size_max=45, range_x=[1,101], range_y=[0,1])
fig.show()
```

[30]:
```
##Encoding Categorical variables
```

[31]:
```
isPopular= pd.get_dummies(dataset['isPopular'], drop_first=True)
isExplicit= pd.get_dummies(dataset['isExplicit'], drop_first=True)
isLoud= pd.get_dummies(dataset['is_loud'], drop_first=True)
```

```
dataset = pd.concat([dataset, isPopular, isExplicit, isLoud], axis = 1)
dataset.head()
```

[31]:

```
       acousticness  danceability  duration_ms  energy  instrumentalness  key  \
19611        0.4010         0.731       205090   0.573          0.000052    4
19606        0.2210         0.700       140526   0.722          0.000000    7
19618        0.0112         0.746       199054   0.765          0.000000    6
19608        0.0194         0.935       187541   0.454          0.000000    1
19610        0.4680         0.737       172325   0.802          0.000000    0

       liveness  loudness  mode  popularity  …     isPopular  Not popular  \
19611    0.1130   -10.059     0         100  …  Very Popular            0
19606    0.2720    -3.558     0          99  …  Very Popular            0
19618    0.0936    -4.410     0          97  …  Very Popular            0
19608    0.0824    -7.509     1          96  …  Very Popular            0
19610    0.0931    -4.771     1          96  …  Very Popular            0

       Popular  Very Popular  Yes  Loud Mellow Soft Very Loud Very Soft
19611        0             1    1     0      0    1         0         0
19606        0             1    1     0      0    0         1         0
19618        0             1    0     0      0    0         1         0
19608        0             1    1     1      0    0         0         0
19610        0             1    1     0      0    0         1         0

[5 rows x 31 columns]
```

[32]:

```
dataset.head()
```

[32]:

```
       acousticness  danceability  duration_ms  energy  instrumentalness  key  \
19611        0.4010         0.731       205090   0.573          0.000052    4
19606        0.2210         0.700       140526   0.722          0.000000    7
19618        0.0112         0.746       199054   0.765          0.000000    6
19608        0.0194         0.935       187541   0.454          0.000000    1
19610        0.4680         0.737       172325   0.802          0.000000    0

       liveness  loudness  mode  popularity  …     isPopular  Not popular  \
19611    0.1130   -10.059     0         100  …  Very Popular            0
19606    0.2720    -3.558     0          99  …  Very Popular            0
19618    0.0936    -4.410     0          97  …  Very Popular            0
19608    0.0824    -7.509     1          96  …  Very Popular            0
19610    0.0931    -4.771     1          96  …  Very Popular            0

       Popular  Very Popular  Yes  Loud Mellow Soft Very Loud Very Soft
19611        0             1    1     0      0    1         0         0
19606        0             1    1     0      0    0         1         0
19618        0             1    0     0      0    0         1         0
19608        0             1    1     1      0    0         0         0
```

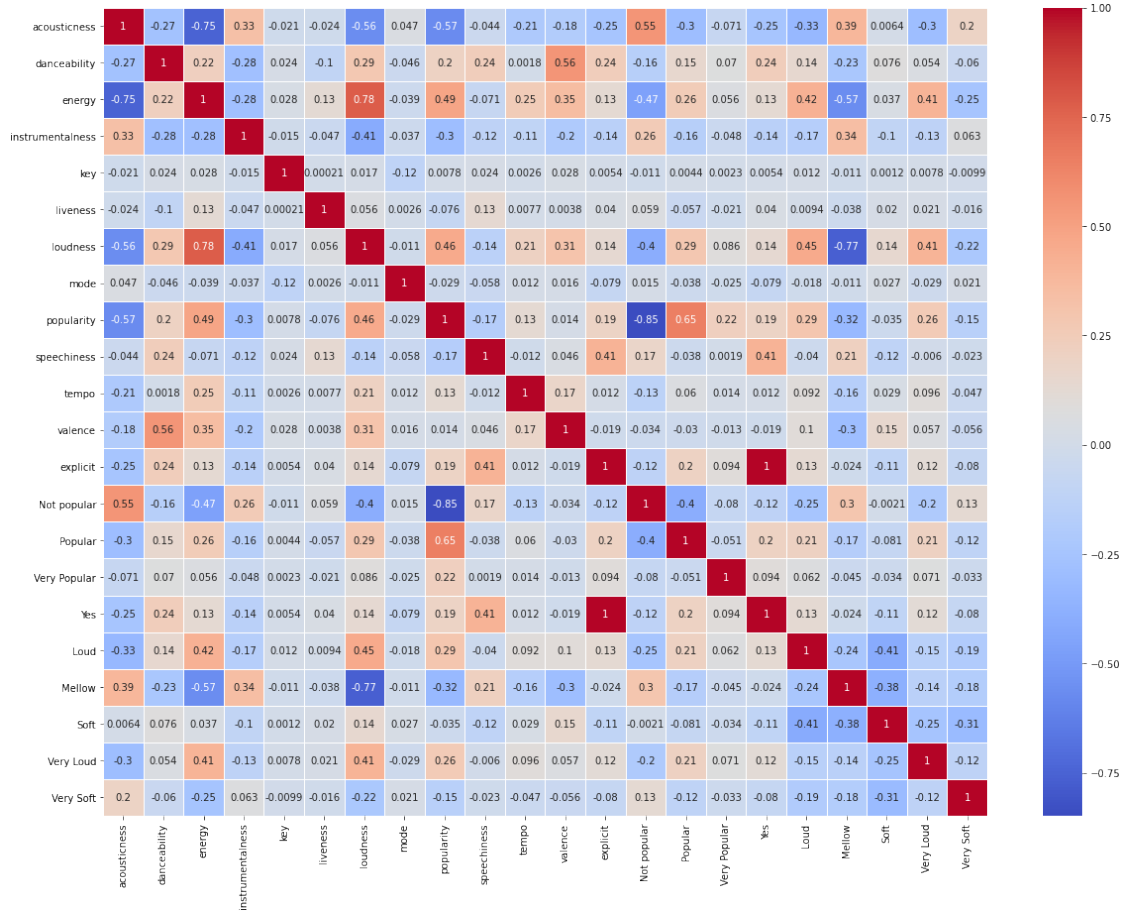
```
19610         0              1     1      0       0     0          1          0

[5 rows x 31 columns]
```

[34]:
```
dataset.drop(['isPopular', 'isExplicit', 'is_loud', 'era', 'en_type',␣
 ↪'artists', 'release_date', 'duration_ms'],
             axis = 1, inplace = True)
```

[36]:
```
dataset.columns
```

[36]:
```
Index(['acousticness', 'danceability', 'energy', 'instrumentalness', 'key',
       'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo',
       'valence', 'explicit', 'year', 'Not popular', 'Popular', 'Very Popular',
       'Yes', 'Loud', 'Mellow', 'Soft', 'Very Loud', 'Very Soft'],
      dtype='object')
```

[68]:
```
plt.figure(figsize=(20,15))
sns.heatmap(dataset.
 ↪corr(),linecolor='white',linewidths=1,cmap='coolwarm',annot=True)
plt.show()
```



[54]:
```
dataset.head()
```

[54]:

	acousticness	danceability	energy	instrumentalness	key	liveness	\
19611	0.4010	0.731	0.573	0.000052	4	0.1130	
19606	0.2210	0.700	0.722	0.000000	7	0.2720	
19618	0.0112	0.746	0.765	0.000000	6	0.0936	
19608	0.0194	0.935	0.454	0.000000	1	0.0824	
19610	0.4680	0.737	0.802	0.000000	0	0.0931	

	loudness	mode	popularity	speechiness	…	year	Not popular	\
19611	-10.059	0	100	0.0544	…	2020	0	
19606	-3.558	0	99	0.0369	…	2020	0	
19618	-4.410	0	97	0.0993	…	2020	0	
19608	-7.509	1	96	0.3750	…	2020	0	
19610	-4.771	1	96	0.0878	…	2020	0	

	Popular	Very Popular	Yes	Loud	Mellow	Soft	Very Loud	Very Soft
19611	0	1	1	0	0	1	0	0
19606	0	1	1	0	0	0	1	0
19618	0	1	0	0	0	0	1	0
19608	0	1	1	1	0	0	0	0
19610	0	1	1	0	0	0	1	0

[5 rows x 23 columns]

[55]:
```
dataset.columns
```

[55]:
```
Index(['acousticness', 'danceability', 'energy', 'instrumentalness', 'key',
       'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo',
       'valence', 'explicit', 'year', 'Not popular', 'Popular', 'Very Popular',
       'Yes', 'Loud', 'Mellow', 'Soft', 'Very Loud', 'Very Soft'],
      dtype='object')
```

[56]:
```
dataset.drop(['year'],axis =1,inplace=True)
```

[38]:
```
##Regression
```

[57]:
```
X= dataset.loc[:,dataset.columns!='popularity']
y= dataset.loc[:,dataset.columns=='popularity']
```

[40]:
```
##Scaling
```

[58]:
```
from sklearn.preprocessing import StandardScaler
sc_X= StandardScaler()
sc_y= StandardScaler()
```

```
X=sc_X.fit_transform(X)
y=sc_y.fit_transform(y)
```

[42]:
```
##Train Split
```

[59]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.40,␣
↪random_state=0)
```

[44]:
```
##Linear Regression
```

[60]:
```
from sklearn.linear_model import LinearRegression
regressor_lin=LinearRegression()
regressor_lin.fit(X_train,y_train)
```

[60]:
```
LinearRegression()
```

[61]:
```
y_pred_lin = regressor_lin.predict(X_test)
```

[62]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

[63]:
```
print("Training Score of Linear Regression is: {}\n".format(regressor_lin.
↪score(X_train, y_train)))
print("R2 Score of Linear Regression is: {}\n".format(r2_score(y_test,␣
↪y_pred_lin)))
print("Mean Squared Error of Linear Regression is: {}\n".
↪format(mean_squared_error(y_test, y_pred_lin)))
print("Mean Absolute Error of Linear Regression is: {}\n".
↪format(mean_absolute_error(y_test, y_pred_lin)))
```

```
Training Score of Linear Regression is: 0.8876972341925911

R2 Score of Linear Regression is: 0.8865360951038146

Mean Squared Error of Linear Regression is: 0.11325345544449593

Mean Absolute Error of Linear Regression is: 0.2863280226562178
```

[64]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
```

[51]:
```
##Random Forest
```

[65]:
```
from sklearn.ensemble import RandomForestRegressor
regressor_rf=RandomForestRegressor(n_estimators=300, random_state=0)
regressor_rf.fit(X_train,y_train)
```

```
y_pred_rf=regressor_rf.predict(X_test)

print("Training Score of Random Forest Regression is: {}\n".format(regressor_rf.
 ↪score(X_train, y_train)))
print("R2 Score of Random Forest Regression is: {}\n".format(r2_score(y_test,␣
 ↪y_pred_rf)))
print("Mean Squared Error of Random Forest Regression is: {}\n".
 ↪format(mean_squared_error(y_test, y_pred_rf)))
print("Mean Absolute Error of Random Forest Regression is: {}\n".
 ↪format(mean_absolute_error(y_test, y_pred_rf)))
```

```
Training Score of Random Forest Regression is: 0.9874782238580833

R2 Score of Random Forest Regression is: 0.9100329534658088

Mean Squared Error of Random Forest Regression is: 0.08980017835148084

Mean Absolute Error of Random Forest Regression is: 0.2405832586466108
```

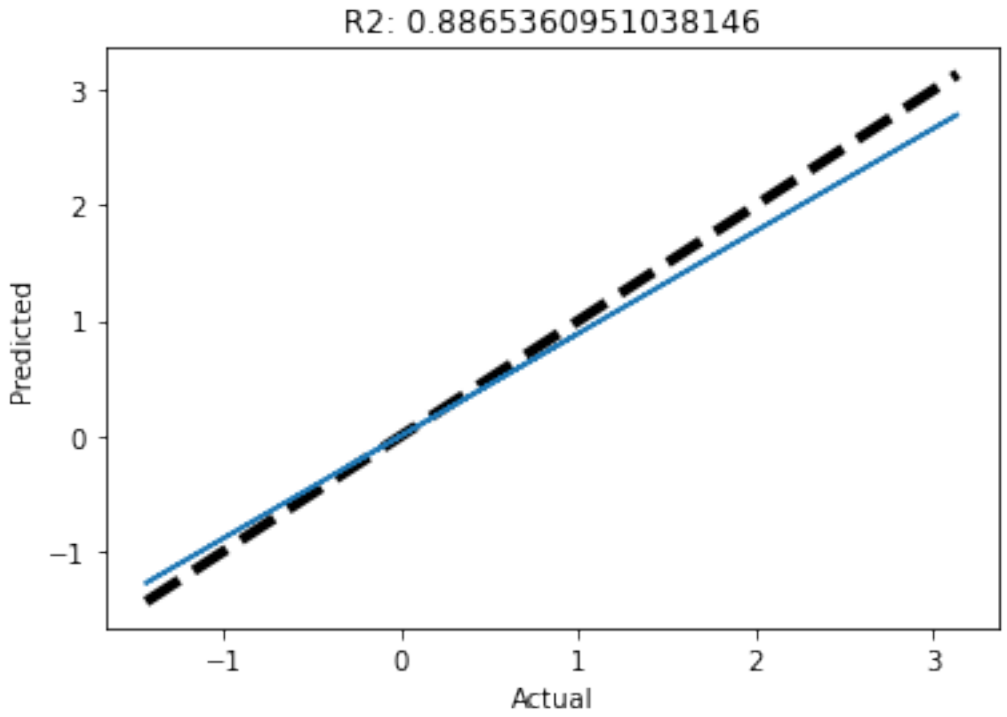
[ ]:
```
##Decision Tree
```

[66]:
```
from sklearn.tree import DecisionTreeRegressor
regressor_dt= DecisionTreeRegressor(random_state=0)
regressor_dt.fit(X_train,y_train)

y_pred_dt= regressor_dt.predict(X_test)

print("Training Score of Decision Tree Regressor is: {}\n".format(regressor_dt.
 ↪score(X_train, y_train)))
print("R2 Score of Decision Tree Regressor is: {}\n".format(r2_score(y_test,␣
 ↪y_pred_dt)))
print("Mean Squared Error of Decision Tree Regressor is: {}\n".
 ↪format(mean_squared_error(y_test, y_pred_dt)))
print("Mean Absolute Error of Decision Tree Regressor is: {}\n".
 ↪format(mean_absolute_error(y_test, y_pred_dt)))
```

```
Training Score of Decision Tree Regressor is: 0.9994531492213072

R2 Score of Decision Tree Regressor is: 0.822054261182486

Mean Squared Error of Decision Tree Regressor is: 0.17761569039199138

Mean Absolute Error of Decision Tree Regressor is: 0.31322498999736176
```

1 Linear Regression

[67]:
```
fig, ax = plt.subplots()
#ax.scatter(y_test, y_pred_lin)
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
#regression line
y_test, y_predicted = y_test.reshape(-1,1), y_pred_lin.reshape(-1,1)
ax.plot(y_test, LinearRegression().fit(y_test, y_pred_lin).predict(y_test))
ax.set_title('R2: ' + str(r2_score(y_test, y_predicted)))
plt.show()
```



[ ]: