Corellation in Python. MovieProject

February 1, 2022

[ ]:
```
# Import the packages we will use in this project
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)

pd.options.mode.chained_assignment = None



# Now we need to read in the data
df = pd.read_csv(r'/Users/andyslo/Desktop/DEV2')
# Now let's take a look at the data

df

# We need to see if we have any missing data

# Let's loop through the data and see if there is anything missing

for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))

# Data Types for our columns

print(df.dtypes)

# Are there any Outliers?
```

```
df.boxplot(column=['gross'])
<AxesSubplot:>

# Order our Data a little bit to see

df.sort_values(by=['gross'], inplace=False, ascending=False)

sns.regplot(x="gross", y="budget", data=df)
<AxesSubplot:xlabel='gross', ylabel='budget'>

sns.regplot(x="score", y="gross", data=df)
<AxesSubplot:xlabel='score', ylabel='gross'>

# Correlation Matrix between all numeric columns

df.corr(method ='pearson')

correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Numeric Features")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()


# Using factorize - this assigns a random numeric value for each unique␣
 ↪categorical value

df.apply(lambda x: x.factorize()[0]).corr(method='pearson')

correlation_matrix = df.apply(lambda x: x.factorize()[0]).corr(method='pearson')

sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Movies")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()
```

```
2

correlation_mat = df.apply(lambda x: x.factorize()[0]).corr()

corr_pairs = correlation_mat.unstack()

print(corr_pairs)



sorted_pairs = corr_pairs.sort_values(kind="quicksort")

print(sorted_pairs)



# We can take a look at the ones that have a high correlation (> 0.5)

strong_pairs = sorted_pairs[abs(sorted_pairs) > 0.5]

print(strong_pairs)

# Looking at the top 15 compaies by gross revenue

CompanyGrossSum = df.groupby('company')[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values('gross', ascending =␣
 ↪False)[:15]

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')

CompanyGrossSumSorted

company



df['Year'] = df['released'].astype(str).str[:4]
df

df.groupby(['company', 'year'])[["gross"]].sum()
gross
company        year
```

```
CompanyGrossSum = df.groupby(['company', 'year'])[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company','year'],␣
↪ascending = False)[:15]

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')

CompanyGrossSumSorted
company                                 year



CompanyGrossSum = df.groupby(['company'])[["gross"]].sum()

CompanyGrossSumSorted = CompanyGrossSum.sort_values(['gross','company'],␣
↪ascending = False)[:15]

CompanyGrossSumSorted = CompanyGrossSumSorted['gross'].astype('int64')






plt.scatter(x=df['budget'], y=df['gross'], alpha=0.5)
plt.title('Budget vs Gross Earnings')
plt.xlabel('Gross Earnings')
plt.ylabel('Budget for Film')
plt.show()








df

df_numerized = df


for col_name in df_numerized.columns:
    if(df_numerized[col_name].dtype == 'object'):
        df_numerized[col_name]= df_numerized[col_name].astype('category')
        df_numerized[col_name] = df_numerized[col_name].cat.codes
```

```
df_numerized

df_numerized.corr(method='pearson')



# Notice. Very interesting marrix

correlation_matrix = df_numerized.corr(method='pearson')

sns.heatmap(correlation_matrix, annot = True)

plt.title("Correlation matrix for Movies")

plt.xlabel("Movie features")

plt.ylabel("Movie features")

plt.show()






for col_name in df.columns:
    if(df[col_name].dtype == 'object'):
        df[col_name]= df[col_name].astype('category')
        df[col_name] = df[col_name].cat.codes




df[cat_columns] = df[cat_columns].apply(lambda x: x.cat.codes)

df



sns.swarmplot(x="rating", y="gross", data=df)


sns.stripplot(x="rating", y="gross", data=df)
<AxesSubplot:xlabel='rating', ylabel='gross'>
```