

## 1) preprocessing Techniques

- **Acquire the dataset:**  
To build and develop Machine Learning models, you must first acquire the relevant dataset. This dataset will be comprised of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. Dataset formats differ according to use cases. For instance, a business dataset will be entirely different from a medical dataset. While a business dataset will contain relevant industry and business data, a medical dataset will include healthcare-related data.

- **Import all the crucial libraries**

we'll show you how to import Python libraries for data preprocessing in Machine Learning:  
Numpy, Pandas, Matplotlib, sklearn

```
import numpy as np  
import pandas as pd  
from sklearn.preprocessing import PolynomialFeatures,  
LabelEncoder
```

- **Import the Dataset**  
**Data = pd.read\_csv('movies\_training.csv')**

- **Extract the independent variables**  
**X = data.iloc[:,1:12]**  
**Y = data['IMDb']**

- **Identifying and handling the missing values**

1- Drop Columns and Rows:

```
data.dropna(axis = 0 ,how='any', inplace=True)
```

```
data.drop('Age',axis=1, inplace=True)
```

2- Calculating the mean

```
Z = Z.groupby(Z.columns, axis = 1).transform(lambda x:  
x.fillna(x.mean()))
```

- Encoding the categorical data

**for c in cols:**

```
lbl = LabelEncoder()
```

```
lbl.fit(list(X[c].values))
```

```
X[c] = lbl.transform(list(X[c].values))
```

- Hot Encoding

```
A= data['Title']
```

```
B= X['Genres']
```

```
# treat null values
```

```
X['Genres'].fillna(' ', inplace = True)
```

```
# separate all genres into one list, considering comma + space  
as separators
```

```
genre = X['Genres'].str.split(',')
```

```
# flatten the list
```

```
flat_genre = [item for sublist in genre for item in sublist]
```

```
# convert to a set to make unique
```

```
set_genre = set(flat_genre)
```

```
# back to list
```

```
unique_genre = list(set_genre)
```

```
# create columns by each unique genre
```

```
Z= X.reindex(X.columns.tolist() + unique_genre, axis=1,  
fill_value=0)
```

```
# for each value inside column, update the dummy
```

```
for index, row in Z.iterrows():
```

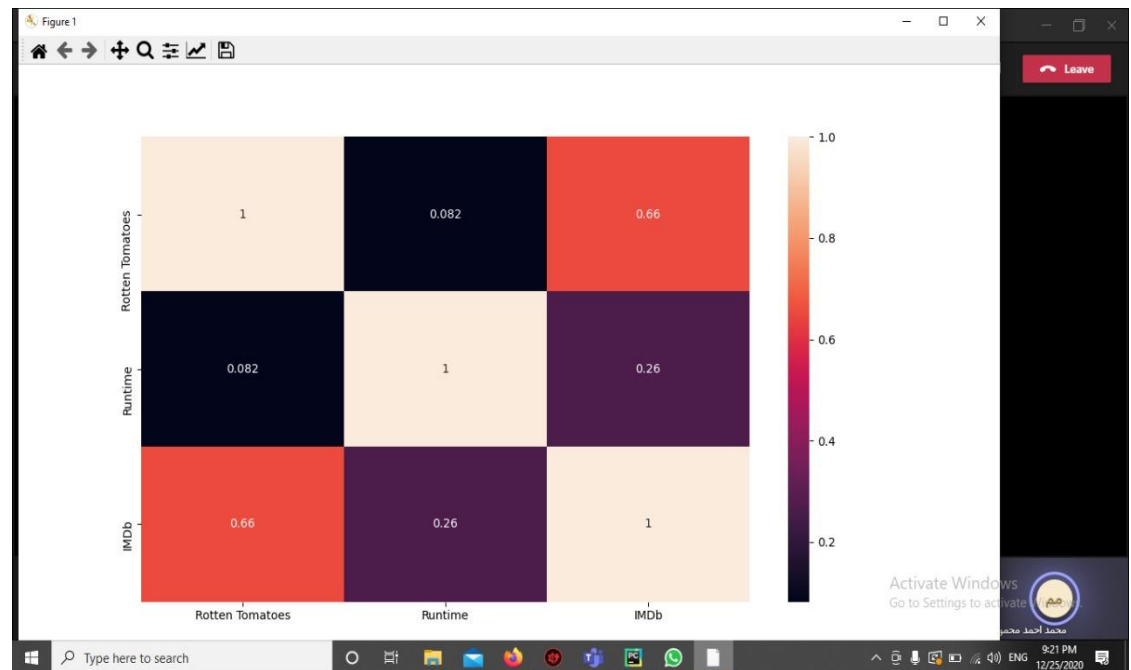
```
for val in row.Genres.split(', '):
```

```
if val != ' ':
```

```
    Z.loc[index, val] = 1
```

```
Z.drop('Genres', axis = 1, inplace = True)
```

2)



After analysis become Top Features are Rotten Tomatoes and Run time, And Age column has low correlation and type column has only zeros in dataset so, I remove them

### 3) Regression Techniques

First Technique : we use multiple linear regression

multiple Linear regression is a basic and commonly used type of predictive analysis which usually works on continuous data. Multiple Linear regressions are based on the assumption that there is a linear relationship between both the dependent and independent variables or Predictor variable and Target variable. It also assumes that there is no major correlation between the independent variables. Multi Linear regressions can be linear and nonlinear. It has one y and two or more x variables or one dependent variable and two or more independent variables.

Second Technique : we use polynomial regression

Polynomial Regression is a one of the types of linear regression in which the relationship between the independent variable  $x$  and dependent variable  $y$  is modeled as an  $n$ th degree polynomial.

Polynomial regression fits a nonlinear relationship between the value of  $x$  and the corresponding conditional mean of  $y$ , denoted  $E(y|x)$ . Polynomial Regression provides the best approximation of the relationship between the dependent and independent variable.

4) difference between multiple linear regression and polynomial regression  
there is no major correlation between the independent variables in multiple linear regression. if we know that our data is correlated, but the relationship doesn't look linear? So hence depending on what the data looks like, we can do a polynomial regression on the data to fit a polynomial equation to it.

Result :

1) Multiple linear regression

MSE : 0.43

Intercept : 24.33

training time : 0.05 second

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help ML - linear regression.py - PyCharm
ML linear regression.py
Run: linear regression
C:\Users\MAYMN\anaconda3\python.exe "C:/Users/MAYMN/Desktop/machine learning/lab/ML/linrear regression.py"
Co-efficient of linear regression [-1.11014698e-02 2.04211801e+00 2.05808767e-01 1.51205244e-01
4.03686327e-02 2.27623042e-01 -1.52952467e-05 1.07471936e-04
6.72213496e-04 -6.38346424e-02 -7.83260459e-03 -1.59406127e-02
7.89753490e-02 1.80521926e-01 3.19597405e-01 6.16537305e-02
2.31620000e-01 -7.00152091e-02 7.85767330e-02 -6.58820869e-02
1.40646832e-02 -2.45444364e-01 9.81677296e-02 -1.53412839e-01
4.15830596e-02 4.41387182e-01 1.22726388e-01 -4.94301192e-01
-7.99587939e-03 -9.62958731e-02 -2.83823337e-01 2.58313828e-01
-6.59614604e-02]
Intercept of linear regression model 27.171627199001577
Mean Square Error 0.46379251105230274
training time 0.050569499999999934 second
True IMdb is : 7.6
Predicted IMdb is : 7.204298791538246
Process finished with exit code 0
Activate Windows
Go to Settings to activate Windows.
18:1 CRLF UTF-8 4 spaces Python 3.8
Type here to search 10:58 PM 12/25/2020
```

## 2) Polynomial regression

MSE : 0.55

Intercept : -1460.4

Training Time : 0.3

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help ML - polynomial regression.py - PyCharm
ML polynomial regression.py
Run: polynomial regression
0.00000000e+00 0.00000000e+00 2.55769204e-03 0.00000000e+00
0.00000000e+00 0.00000000e+00 1.54626503e+00 1.82301221e-02
-1.75277316e-01 -2.91470504e-01 -1.22475977e-01 -2.18471848e-01
4.04716684e-01 -8.37874334e-02 -1.39782798e-01 -1.24799464e+01
0.00000000e+00 -1.12263586e-01 1.06453191e-02 -2.65622210e-01
-4.15990219e-02 -3.56498102e-01 8.67532511e-01 1.52121563e-04
1.17714785e-01 0.00000000e+00 -4.49039649e-02 0.00000000e+00
3.37896065e-01 -1.75277316e-01 -3.52894849e+00 -3.13647532e-02
-9.05991700e-02 -8.11030708e-02 -3.06673003e-01 4.53211278e-02
-8.73258650e+00 3.05849103e-01 -4.13109026e-01 -5.22974200e-01
8.62519338e-02 -2.55319515e-01 2.29472576e-01 0.00000000e+00
-8.63170069e-02 -2.25891712e+00 -1.23981146e-01 -1.55592566e-01
-9.00518348e+00 4.85265927e-02 -1.37523496e+01]
Intercept of polynomial regression model -1498.9742816410878
Mean Square Error 0.589765777755048
training time 0.305042500000000077 second
True IMdb is : 6.2
Predicted IMdb is : 6.367805479236495
Process finished with exit code 0
Activate Windows
Go to Settings to activate Windows.
15:1 CRLF UTF-8 4 spaces Python 3.8
Type here to search 11:01 PM 12/25/2020
```

5) You must clearly mention what features you used or discarded to create your regression models.

( Year, Netflix , Hulu, Prime Video , Disney+ , Type , Directors , Genres  
Country , Language , Runtime )

6) sizes of your training, testing

Training : 70%

Testing: 30%

8) You should include screenshots of the resultant(s) regression line plots if possible or any data visualization.

9) at first we remove Rotten Tomatoes column and  $MSE > 1$  but when we edit that column and add mean values in nan cells  $MSE < 1$  almost 0.4

