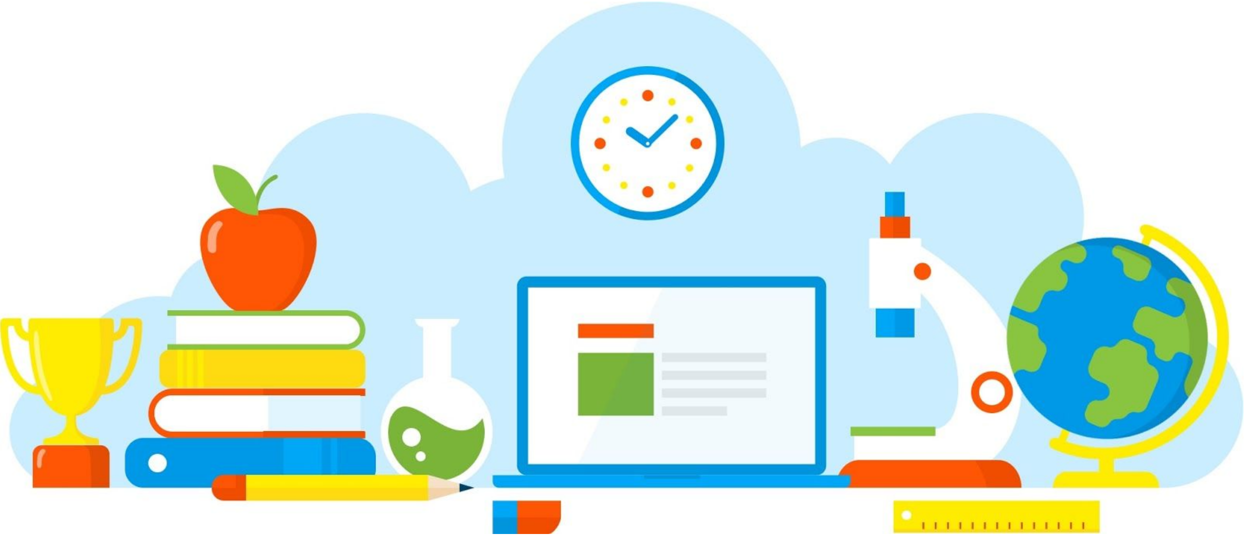


Milestone_1 Report

Predict Movie Success
25/12/2020

Pre_processing

First, we read data from a csv file .Then decide to drop the column that has a constant value . Then takes care about missing value by drop the rows with none data in some columns that is important in our data like:
'IMDb','Runtime','Country','Language','Directors','Genres','Year'

Then fill the data of nan value in 'Age' ,'Rotten Tomatoes by random value from its data

After all that the date become (10599, 15) from (11744 ,15)

Then, make Label Encoding to the get a numerical value to the string one on all string data except the chosen categorical column which is 'Genres' to choose the categorical column try the Country','Language but it gets a higher MSE so we decided to choose a 'Genres' to make on it one hot_encoding manually to get the each genres of movies in one column by

```
GF = df1.Genres.str.split(r'\s*,\s*', expand=True).apply(pd.Series.value_counts, 1) .iloc[:, 1:].fillna(0,
downcast='infer')
```

df = pd.concat([df1, GF.reindex(df.index)], axis=1, join='inner') and try w different columns with different models of regression (details in regression section).

Then normalize the data by scaling to make it with value between (0,1) to make the feathers have the same value to the model

Check if the “IMDb” is the last column or not and if it not make it the last

Then checked on the data to delete any row contain nan value df= df.dropna(0)

So, we have a cleaned data after the preprocessing phase and saved it in a csv to don't make this phase each time of training in several model

The feature (36) uses:

year	Age	Rotten t	Director	Runtime	Country	Netflix	Hulu	Prime Video	Disney+	**Genres**

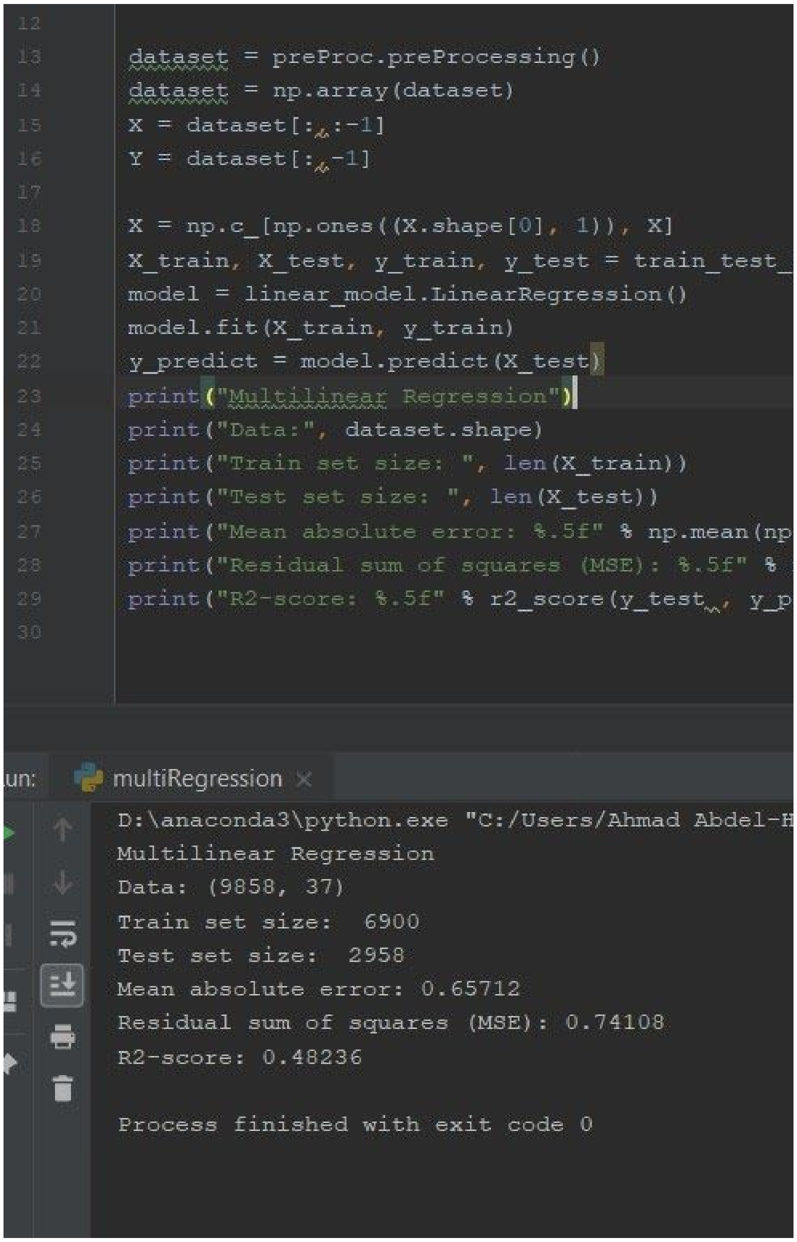
Ad	An	Bi	Co	Cri	Do	Dr	Fa	Fa	Fil	Hi	Ho	Mu	Mu	My	Ne	Re	Ro	Sci	Sh	Sp	Tal	Th	W	W

Regression

MultiRegression

It is a first regression technique we use, tried it on different shape of data

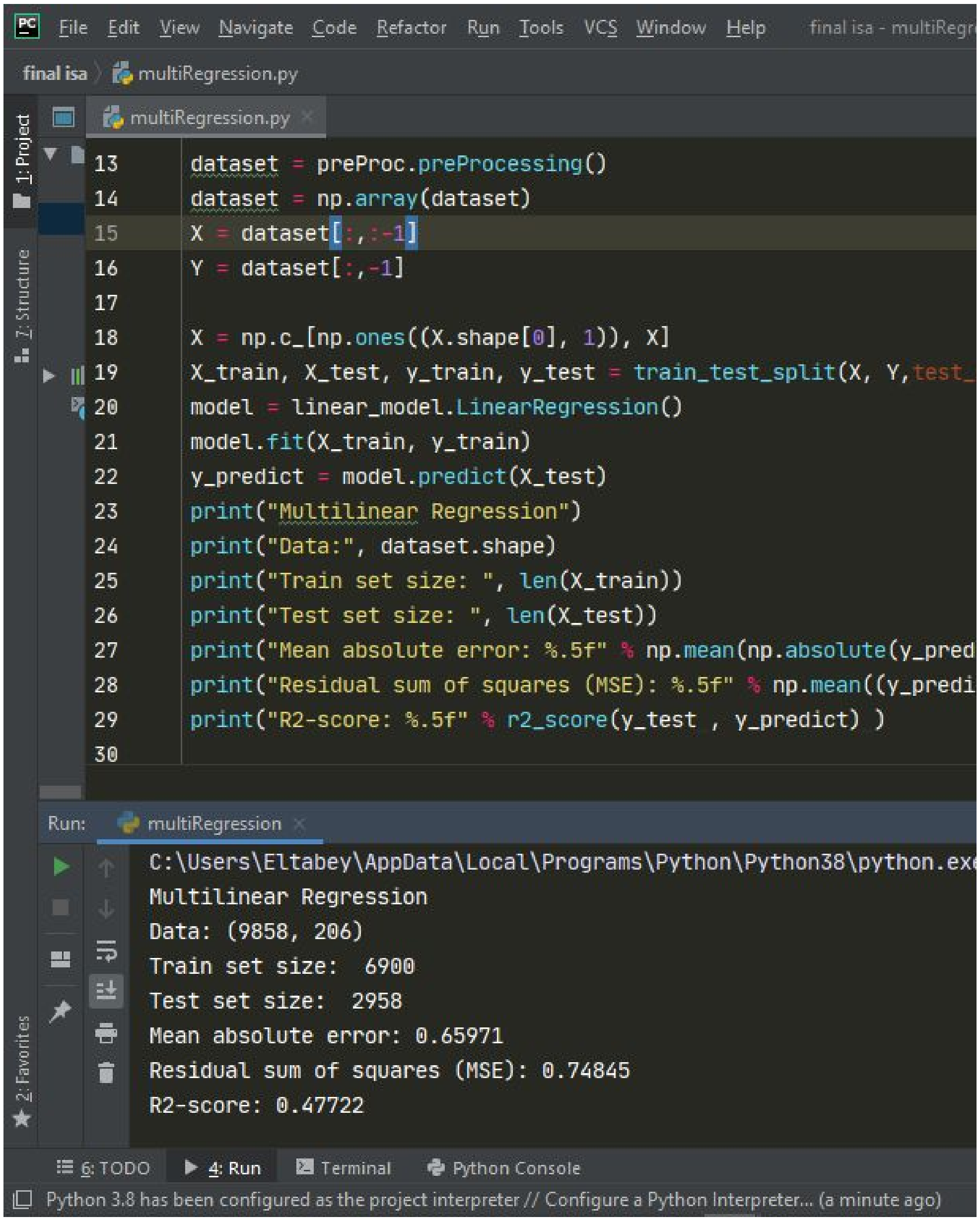
- The data is (9858, 37) which the categorical feature is ‘’Genres’’ and it was the best MSE
- Less than 1 minutes to train



```
dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]

X = np.c_[np.ones((X.shape[0], 1)), X]
X_train, X_test, y_train, y_test = train_test_
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print("Multilinear Regression")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
print("Mean absolute error: %.5f" % np.mean(np
print("Residual sum of squares (MSE): %.5f" % 
print("R2-score: %.5f" % r2_score(y_test, y_p
```

```
D:\anaconda3\python.exe "C:/Users/Ahmad Abdel-H
Multilinear Regression
Data: (9858, 37)
Train set size:  6900
Test set size:  2958
Mean absolute error: 0.65712
Residual sum of squares (MSE): 0.74108
R2-score: 0.48236

Process finished with exit code 0
```

- When we make more than one column (Country, language , Genres)as categorical features we gat that the data was huge (9858,206) , train size = 6900, tasting set size = 2958
- Takes less 2 minutes to train



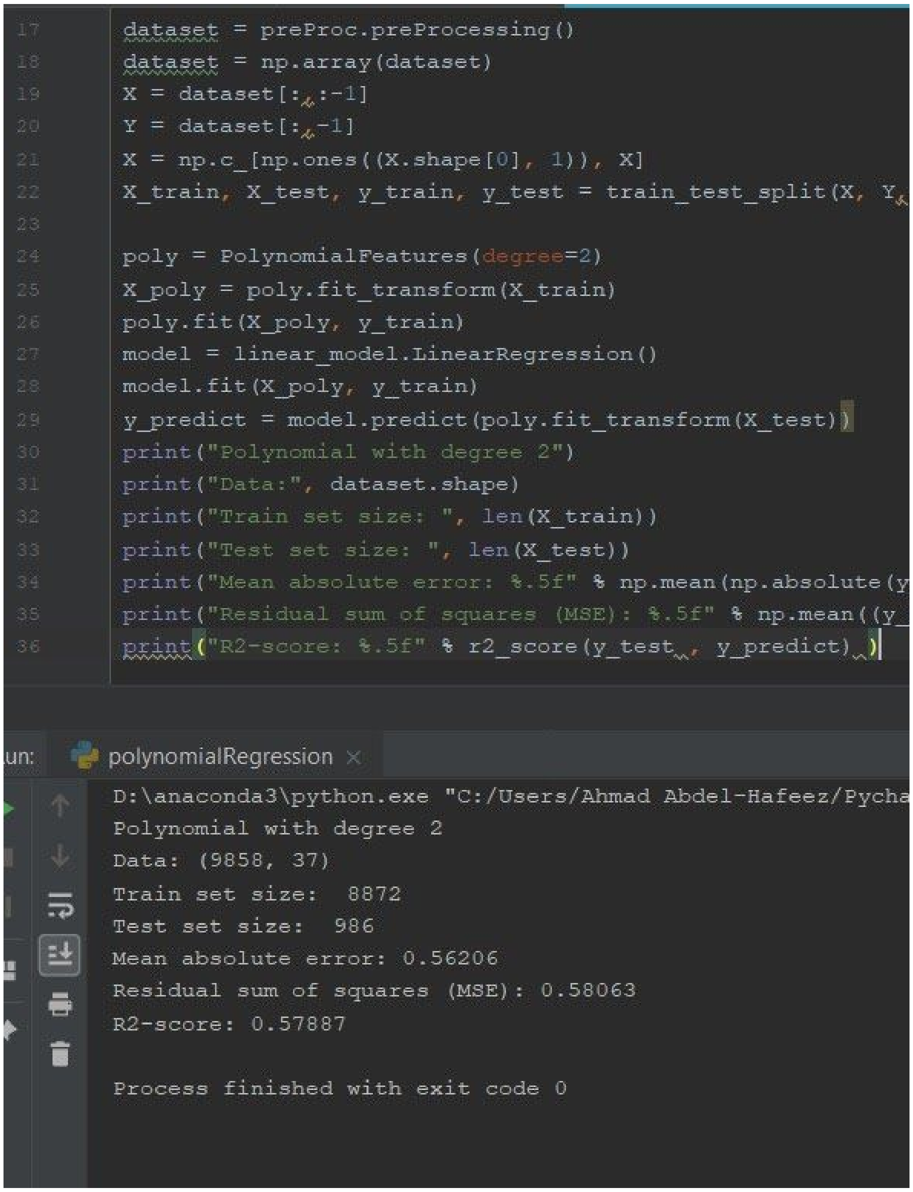
```python
dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]

X = np.c_[np.ones((X.shape[0], 1)), X]
X_train, X_test, y_train, y_test = train_test_split(X, Y,test_
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
print("Multilinear Regression")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
print("Mean absolute error: %.5f" % np.mean(np.absolute(y_pred
print("Residual sum of squares (MSE): %.5f" % np.mean((y_predi
print("R2-score: %.5f" % r2_score(y_test , y_predict) )
```

```
C:\Users\Eltabey\AppData\Local\Programs\Python\Python38\python.exe
Multilinear Regression
Data: (9858, 206)
Train set size:  6900
Test set size:  2958
Mean absolute error: 0.65971
Residual sum of squares (MSE): 0.74845
R2-score: 0.47722
```

Polynomial Regression

It is a second regression technique we use, tried it on different shape of data

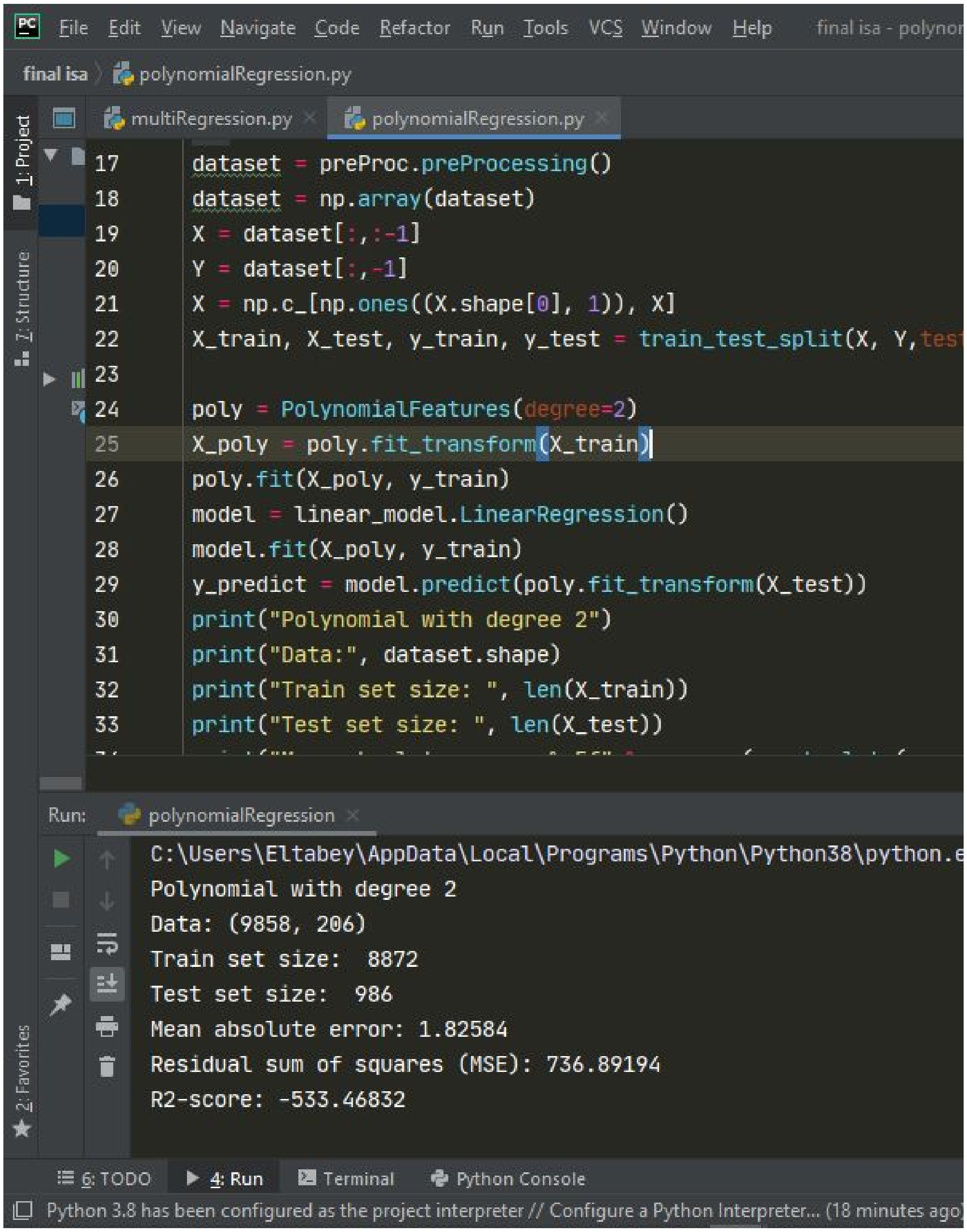
- The data is (9858, 37) which the categorical feature is “Genres” and it was the best MSE , train size set = (8872) , test size set = (986) .
- Takes less than 1 minute to train
- Its a polynomial of degree 2 that is doesn't take much time if we increase the degree it is more efficient but takes much time



```
dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]
X = np.c_[np.ones((X.shape[0], 1)), X]
X_train, X_test, y_train, y_test = train_test_split(X, Y,

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)
poly.fit(X_poly, y_train)
model = linear_model.LinearRegression()
model.fit(X_poly, y_train)
y_predict = model.predict(poly.fit_transform(X_test))
print("Polynomial with degree 2")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
print("Mean absolute error: %.5f" % np.mean(np.absolute(y
print("Residual sum of squares (MSE): %.5f" % np.mean((y_
print("R2-score: %.5f" % r2_score(y_test, y_predict))
```

```
D:\anaconda3\python.exe "C:/Users/Ahmad Abdel-Hafeez/Pycha
Polynomial with degree 2
Data: (9858, 37)
Train set size:  8872
Test set size:  986
Mean absolute error: 0.56206
Residual sum of squares (MSE): 0.58063
R2-score: 0.57887

Process finished with exit code 0
```

- When we make more than one column as categorical features we gat that the data was huge (9858,206) , train size = 6900, tasting set size = 2958
- Takes 20 mints and doesn't give any efficient value



```python
dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]
X = np.c_[np.ones((X.shape[0], 1)), X]
X_train, X_test, y_train, y_test = train_test_split(X, Y,tes

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)
poly.fit(X_poly, y_train)
model = linear_model.LinearRegression()
model.fit(X_poly, y_train)
y_predict = model.predict(poly.fit_transform(X_test))
print("Polynomial with degree 2")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
```

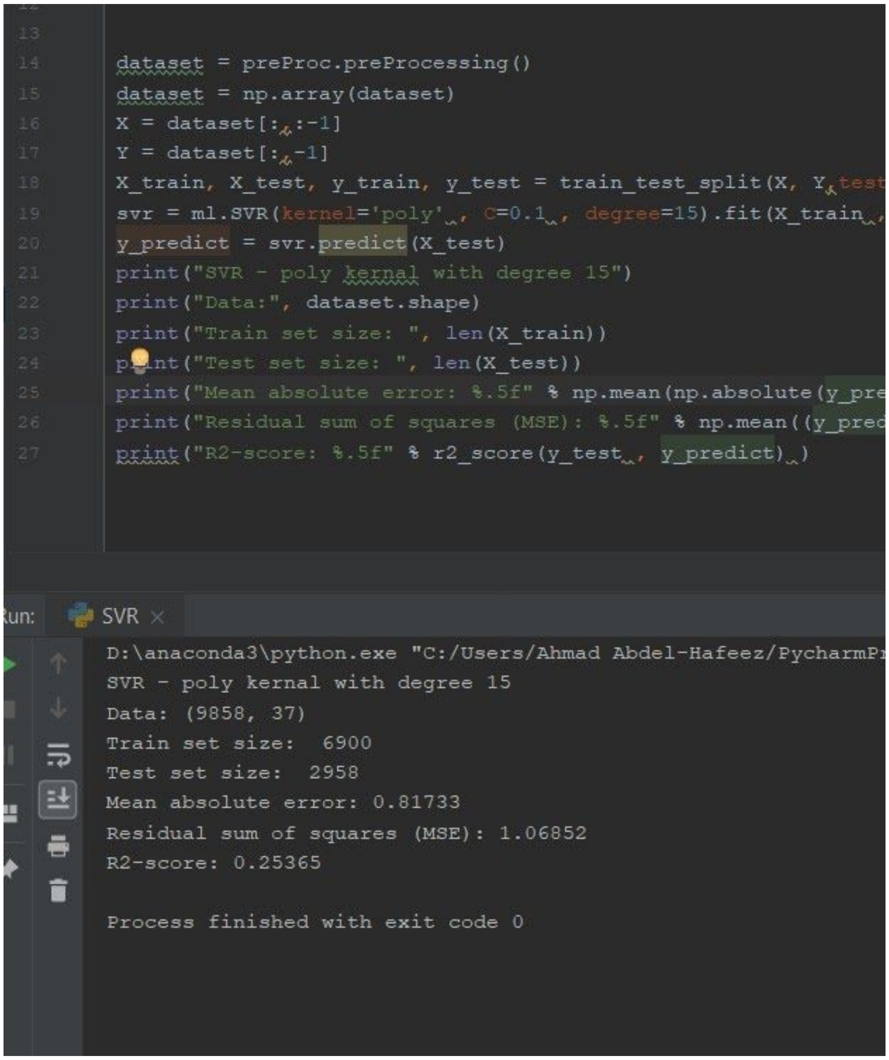
```
C:\Users\Eltabey\AppData\Local\Programs\Python\Python38\python.e
Polynomial with degree 2
Data: (9858, 206)
Train set size:  8872
Test set size:  986
Mean absolute error: 1.82584
Residual sum of squares (MSE): 736.89194
R2-score: -533.46832
```

Python 3.8 has been configured as the project interpreter // Configure a Python Interpreter... (18 minutes ago)

SVR Regression

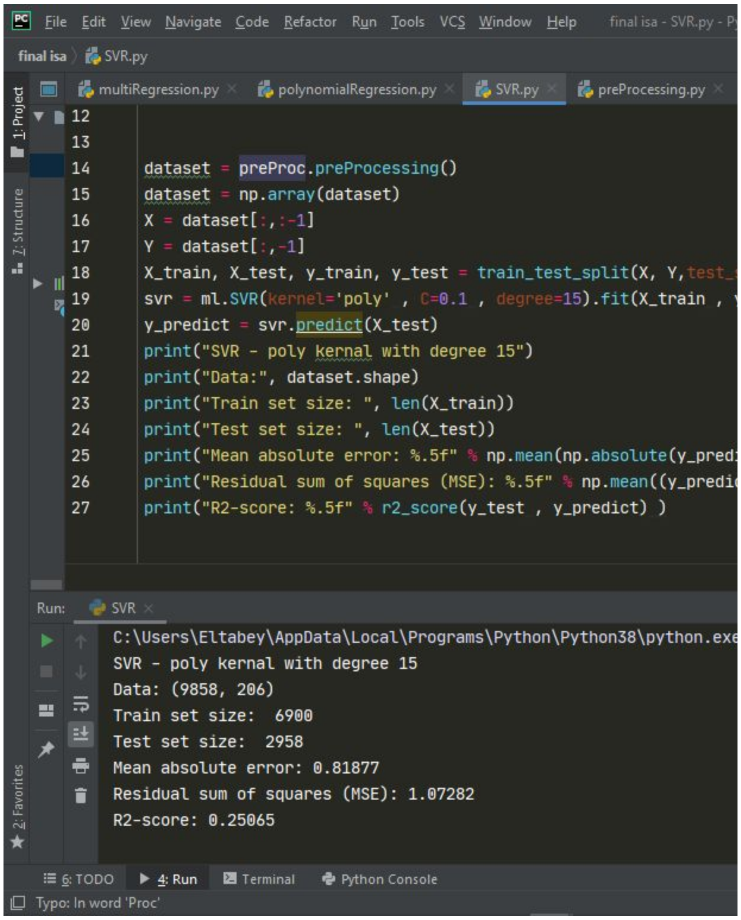
It is a third regression technique we use, tried it on different shape of data

- The data is (9858, 37) which the categorical feature is “Genres” and it was the best MSE , train size set = (8872) , test size set = (986) .
- SVR poly kernel with degree 15
- Takes 10 minutes to train



```
dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, Y,test
svr = ml.SVR(kernel='poly' , C=0.1 , degree=15).fit(X_train ,
y_predict = svr.predict(X_test)
print("SVR - poly kernal with degree 15")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
print("Mean absolute error: %.5f" % np.mean(np.absolute(y_pre
print("Residual sum of squares (MSE): %.5f" % np.mean((y_pred
print("R2-score: %.5f" % r2_score(y_test , y_predict) )
```

```
D:\anaconda3\python.exe "C:/Users/Ahmad Abdel-Hafeez/PycharmPr
SVR - poly kernal with degree 15
Data: (9858, 37)
Train set size:  6900
Test set size:  2958
Mean absolute error: 0.81733
Residual sum of squares (MSE): 1.06852
R2-score: 0.25365

Process finished with exit code 0
```

- When we make more than one column (Country, language , Genres)as categorical features we gat that the data was huge (9858,206) , train size = 6900, tasting set size = 2958 and tries to changes the splitting data ratio to be more efficient
- SVR poly kernel with degree 15
- Takes 13 minutes to train
- 

```


dataset = preProc.preProcessing()
dataset = np.array(dataset)
X = dataset[:,:-1]
Y = dataset[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, Y,test_
svr = ml.SVR(kernel='poly' , C=0.1 , degree=15).fit(X_train , y
y_predict = svr.predict(X_test)
print("SVR - poly kernal with degree 15")
print("Data:", dataset.shape)
print("Train set size: ", len(X_train))
print("Test set size: ", len(X_test))
print("Mean absolute error: %.5f" % np.mean(np.absolute(y_predi
print("Residual sum of squares (MSE): %.5f" % np.mean((y_predic
print("R2-score: %.5f" % r2_score(y_test , y_predict) )
```

```
C:\Users\Eltabey\AppData\Local\Programs\Python\Python38\python.exe
SVR - poly kernal with degree 15
Data: (9858, 206)
Train set size:  6900
Test set size:  2958
Mean absolute error: 0.81877
Residual sum of squares (MSE): 1.07282
R2-score: 0.25065
```

Conclusion

On the final data which its size is (9858, 37) and the categorical feature is "Genres" , the best model is polynomial regression(second degree) as it gives the smallest MSE =0.58.

And on the huge data which its size is (9858,206) and(Country, language , Genres)as categorical features , the best model is multi regression as it gives MSE = 0.74.

By changing the ratio of splitting data that change the size of training set and test set it effects on MSE