American International University Bangladesh

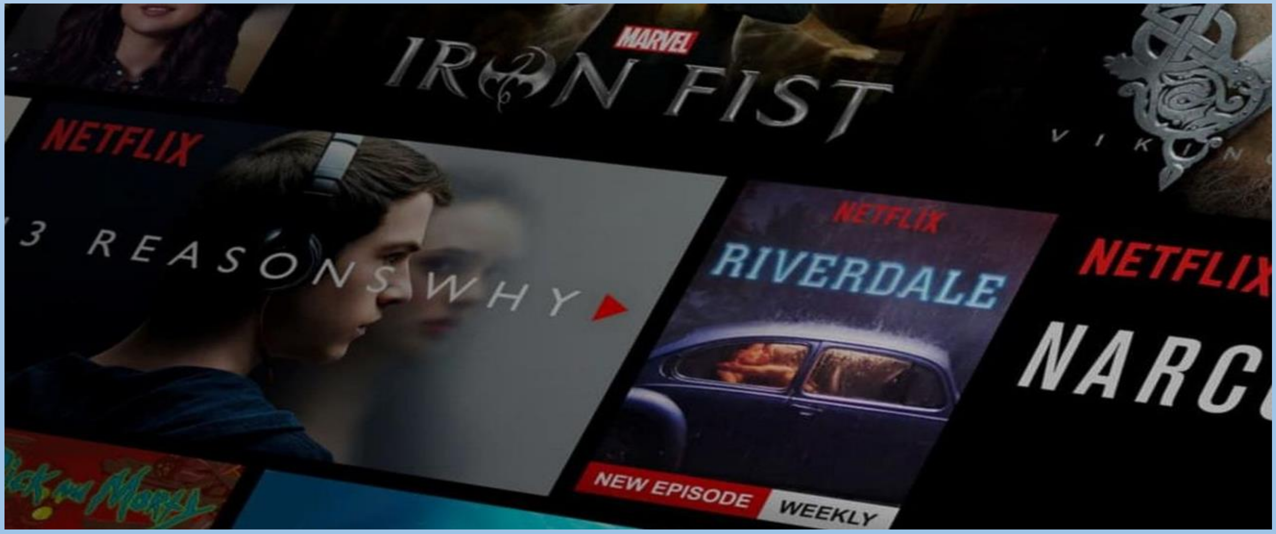
Movie Recommendation System

Group members-

ZISAN AHMED(19-39294-1)
TANVIR HOSSAIN FIROZ(16-32222-2)
MEHEDI HASAN MAHIN(17-34902-2)
TUYAN JAMAMIM(20-43978-2)

Programming In Python

Section :B



Movie Recommendation System

Project overview

The goal of the Movie Recommendation System project is to construct a classification-based machine learning model that will enable users to receive tailored movie recommendations based on their tastes. The project made use of a dataset that came from IMDB ratings and included information on the director's name, budget, Facebook likes, and more.

Data collection, cleaning, and preparation were the first steps in the project's structured methodology. Several missing values and unnecessary columns were present in the dataset, which were eliminated during the cleaning process. To maintain data integrity, rows with high null percentages were also removed.

Exploratory data analysis was done after data cleaning to learn more about the dataset's structure and spot trends and patterns. The analysis was useful in helping to choose the pertinent features for the classification models.

The project used Naive Bayes, KNN, Decision Trees, Logistic Regression, and SVM as five classification models. Each model's predicted accuracy was examined after it had been trained on the cleaned dataset. To find the best successful model, the classifiers were compared.

Project Methodology

Data Collection

Data Source-

We have downloaded the data from https://www.itronixsolutions.com/imdb-movies-data-cleaning-and-data-analysis-using-python/

The project utilized the use of an IMDB dataset that includes information on the director's name, budget, Facebook likes, and more.

```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt


movie= pd.read_csv('IMDB_Movies.csv')

Original_data=movie

movie
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	...	num_user_
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	...	
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0	Action|Adventure|Fantasy	...	
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0	Action|Adventure|Thriller	...	
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0	Action|Thriller	...	
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN	Documentary	...	
...	...	...	...	...	...	...	...	...	...	...	...	
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	NaN	Comedy|Drama	...	
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	841.0	NaN	Crime|Drama|Mystery|Thriller	...	
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	NaN	Drama|Horror|Thriller	...	
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	10443.0	Comedy|Drama|Romance	...	
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	85222.0	Documentary	...	

5043 rows × 28 columns

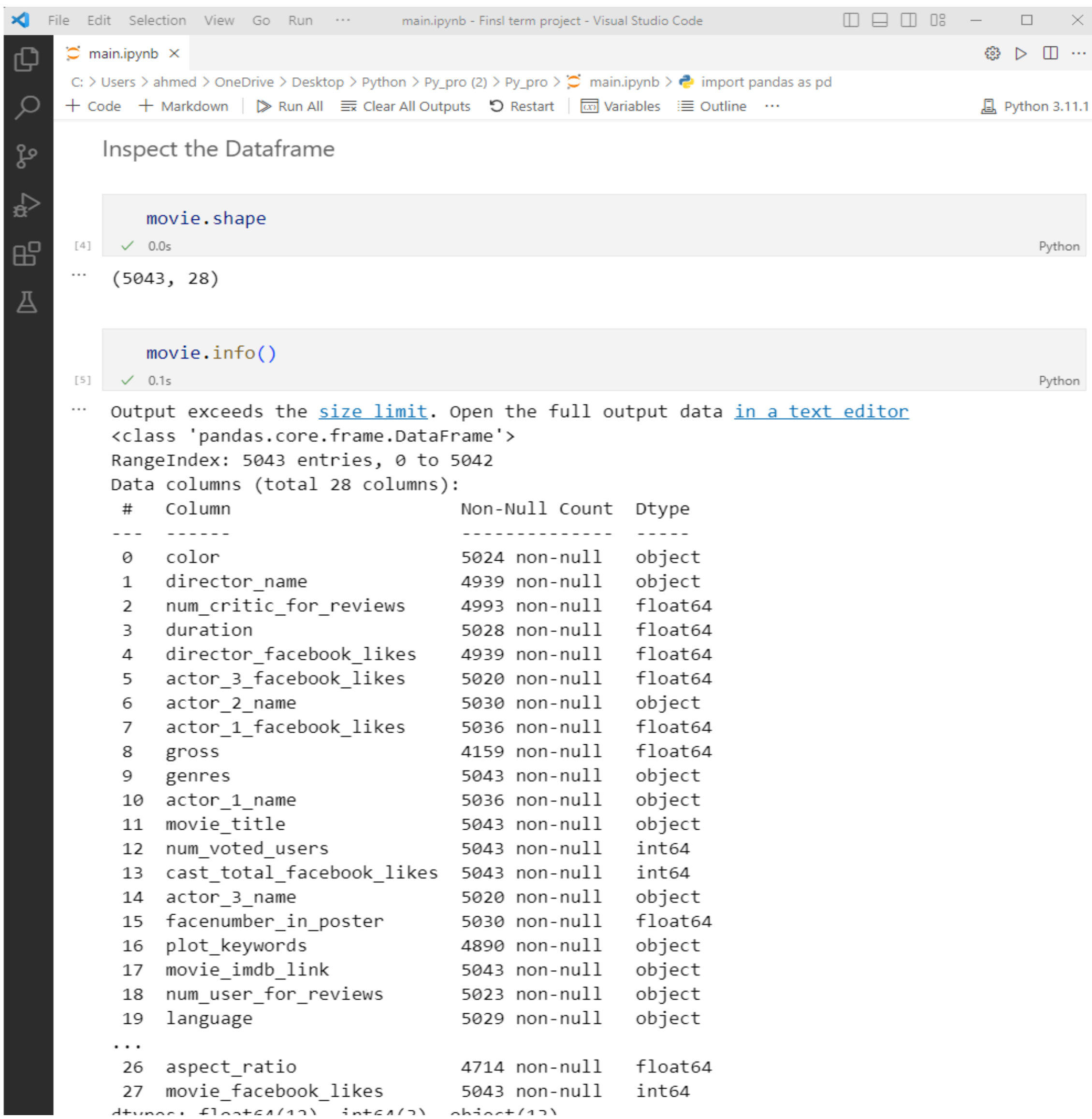
The dataset comprises information on 5043 movies, including the director's name, the number of critic reviews, the length of the film, the gross earnings, genres, and language, among other things.

Some columns in the data have missing values, such as director name, duration, and gross earnings. Aspect ratio and content rating columns, for example, have missing entries as well as a "NaN" value.

The data contains films from several countries, including the United States, the United Kingdom, and Canada, among others. The films are divided into several categories, including Action, Adventure, Fantasy, Comedy, Drama, and Documentary. Movies' gross earnings range from 0 to over 760 million dollars. The number of reviewer reviews varies between one and over 800. IMDb ratings range from 1.6 to 9.5.

There are also categorical factors in the dataset, such as content rating and language. Within each of these variables, there are several levels.

Overall, the dataset contains a wide range of movies and their properties, making it suitable for analysis and modeling. Missing values and categorical variables, on the other hand, may necessitate some preparation before analysis



Inspect the Dataframe

```
movie.shape
```

[4] 0.0s Python

```
(5043, 28)
```

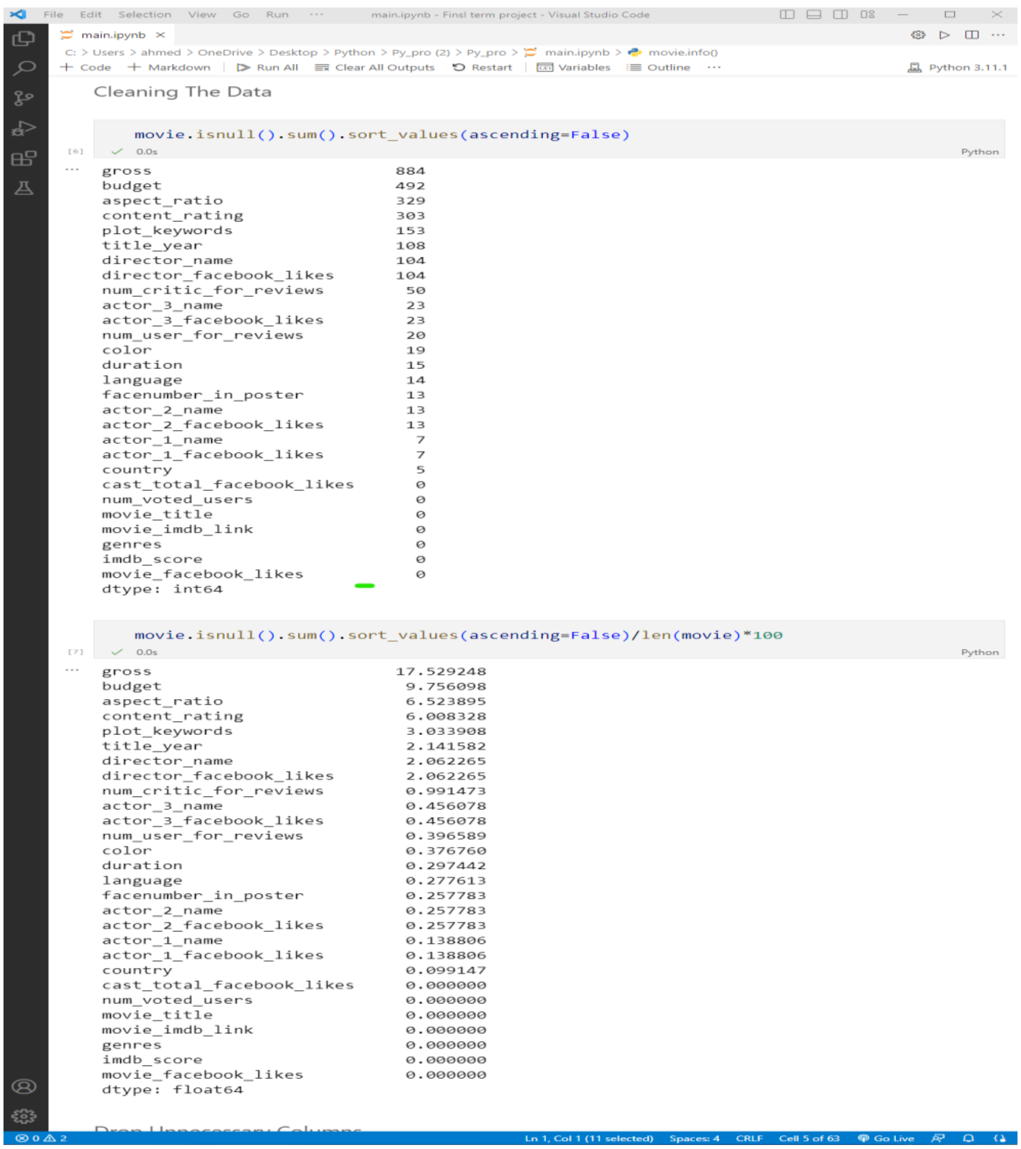
```
movie.info()
```

[5] 0.1s Python

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   color                      5024 non-null   object
 1   director_name              4939 non-null   object
 2   num_critic_for_reviews     4993 non-null   float64
 3   duration                   5028 non-null   float64
 4   director_facebook_likes    4939 non-null   float64
 5   actor_3_facebook_likes     5020 non-null   float64
 6   actor_2_name               5030 non-null   object
 7   actor_1_facebook_likes     5036 non-null   float64
 8   gross                      4159 non-null   float64
 9   genres                     5043 non-null   object
 10  actor_1_name               5036 non-null   object
 11  movie_title                5043 non-null   object
 12  num_voted_users            5043 non-null   int64
 13  cast_total_facebook_likes  5043 non-null   int64
 14  actor_3_name               5020 non-null   object
 15  facenumber_in_poster       5030 non-null   float64
 16  plot_keywords              4890 non-null   object
 17  movie_imdb_link            5043 non-null   object
 18  num_user_for_reviews       5023 non-null   object
 19  language                   5029 non-null   object
...
 26  aspect_ratio               4714 non-null   float64
 27  movie_facebook_likes       5043 non-null   int64
```

The movie dataset comprises information on 5043 movies, each with 28 columns of data. The columns contain information such as the title of the film, its director, actors, genre, gross earnings, duration, and other pertinent statistics.

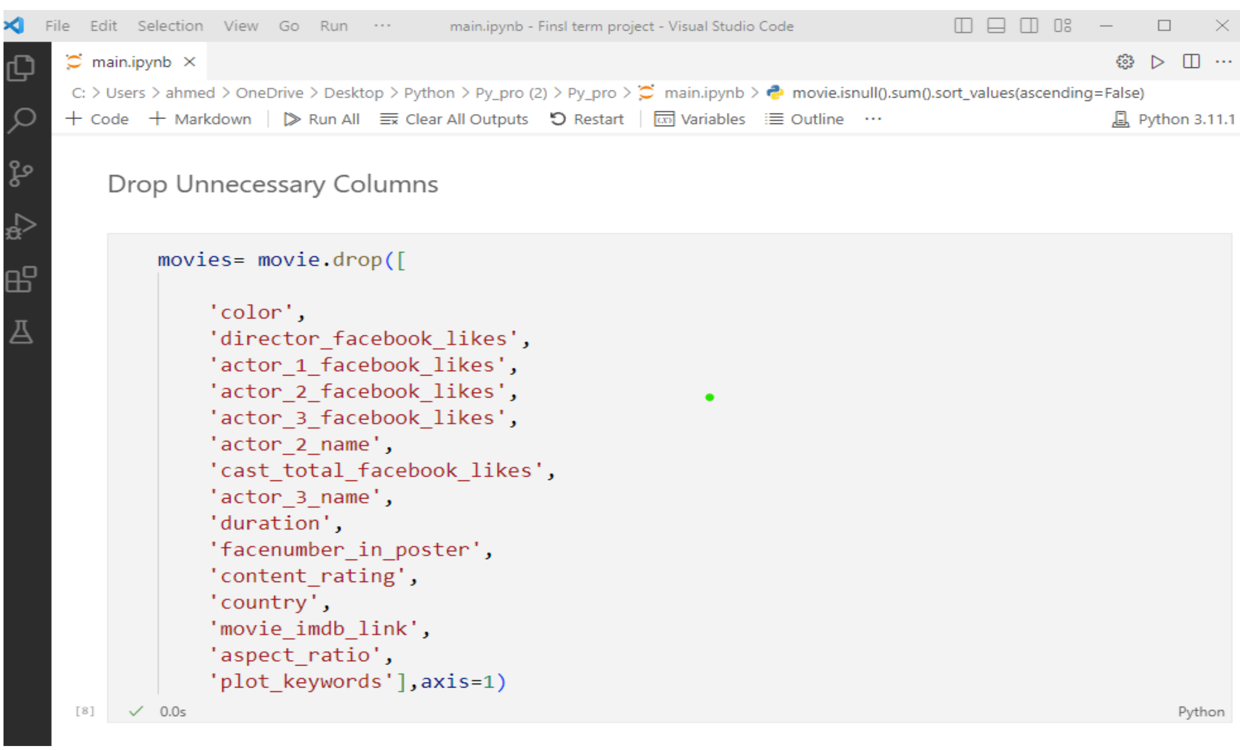
The collection contains a variety of data types, including 12 float64, 3 int64, and 13 object (string). The dataset includes some missing values, as well as varied numbers of non-null items in each column.

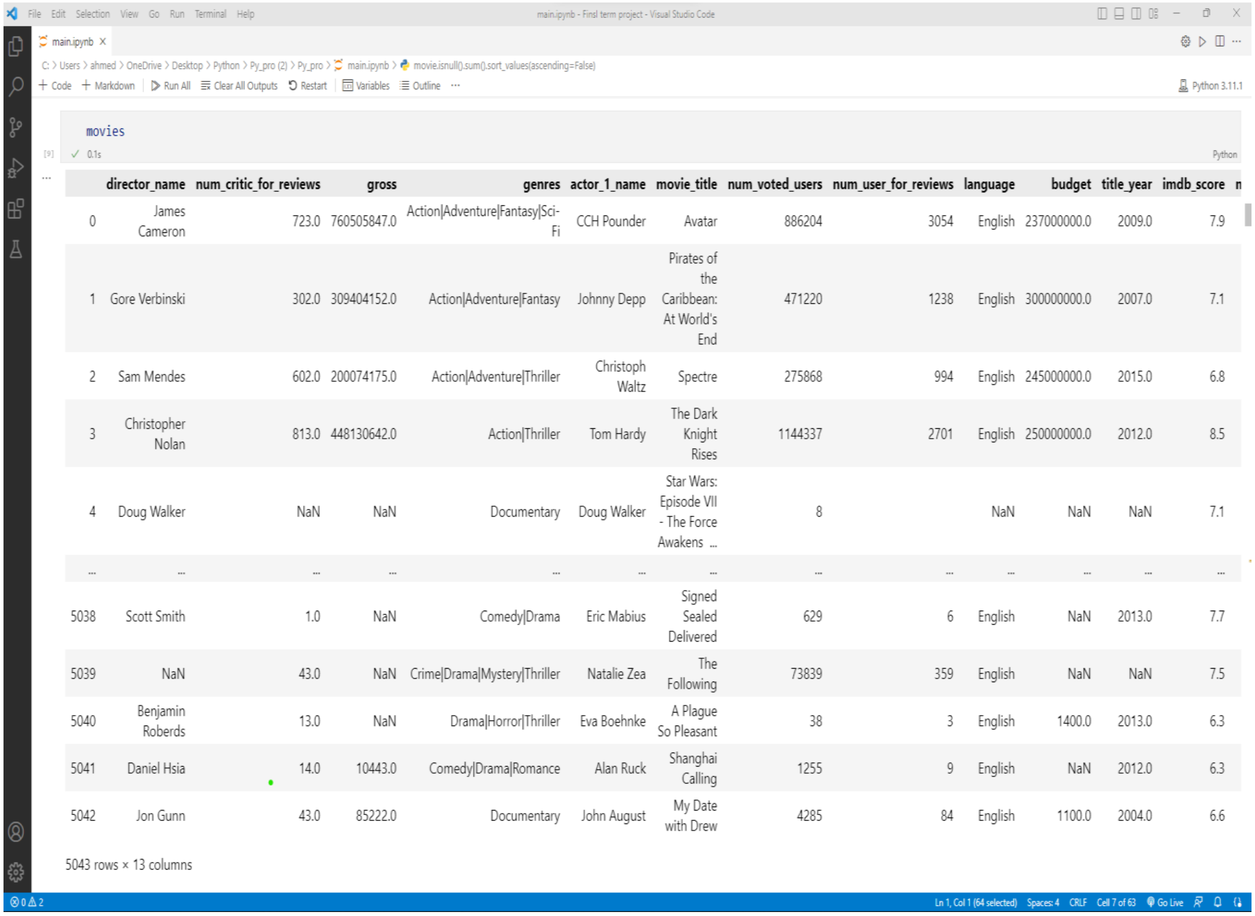


These two lines of code provide information about the movie dataframe's missing values.

The first line of the film.isnull().sum().sort_values(ascending=False) counts the amount of missing values in each dataframe column and arranges them descendingly. This enables us to determine which columns have the greatest number of missing values.

The second line of code is movie.isnull().sum().sort_values(ascending=False)/len(movie)*100. divides the number of missing values by the total number of rows in the dataframe and multiplies by 100 to compute the percentage of missing values in each column. This allows for a more realistic comparison of the amount of missing data across different columns.





	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	imdb_score	n
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	English	237000000.0	2009.0	7.9	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	English	300000000.0	2007.0	7.1	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	English	245000000.0	2015.0	6.8	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	English	250000000.0	2012.0	8.5	
4	Doug Walker	NaN	NaN	Documentary	Doug Walker	Star Wars: Episode VII - The Force Awakens ...	8		NaN	NaN	NaN	7.1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5038	Scott Smith	1.0	NaN	Comedy|Drama	Eric Mabius	Signed Sealed Delivered	629	6	English	NaN	2013.0	7.7	
5039	NaN	43.0	NaN	Crime|Drama|Mystery|Thriller	Natalie Zea	The Following	73839	359	English	NaN	NaN	7.5	
5040	Benjamin Roberds	13.0	NaN	Drama|Horror|Thriller	Eva Boehnke	A Plague So Pleasant	38	3	English	1400.0	2013.0	6.3	
5041	Daniel Hsia	14.0	10443.0	Comedy|Drama|Romance	Alan Ruck	Shanghai Calling	1255	9	English	NaN	2012.0	6.3	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	84	English	1100.0	2004.0	6.6	

5043 rows × 13 columns

We are removing some columns from the original movie dataset in the code above since they do not appear to be important to the study or have a considerable amount of missing information. The omitted columns include information on the movie's color, the amount of Facebook likes for the director and actors, the runtime of the film, the number of faces on the poster, the content rating, the country of production, an IMDB link to the film, the aspect ratio, and storyline keywords.

The following columns will be utilized for analysis and will include information on the director's name, the number of critic reviews, the gross revenues, the genre, the names of the actors, the movie title, the number of users who voted for the movie, the number of user reviews, and the language of the movie.

Drop The Unnecessary Rows Using Columns With High Null Percentage

```python
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

[10] ✓ 0.0s

```
gross                   17.53
budget                   9.76
title_year               2.14
director_name            2.06
num_critic_for_reviews   0.99
num_user_for_reviews     0.40
language                 0.28
actor_1_name             0.14
genres                   0.00
movie_title              0.00
num_voted_users          0.00
imdb_score               0.00
movie_facebook_likes     0.00
dtype: float64
```

```python
movies=movies[movies['gross'].notnull()]
movies=movies[movies['budget'].notnull()]
movies
```

[11] ✓ 0.0s

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	imdb_s
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	English	237000000.0	2009.0	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	English	300000000.0	2007.0	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	English	245000000.0	2015.0	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	English	250000000.0	2012.0	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	738	English	263700000.0	2012.0	
...	...	...	...	...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	371	English	7000.0	2004.0	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	35	English	7000.0	2005.0	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	130	Spanish	7000.0	1992.0	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	14	English	9000.0	2011.0	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	84	English	1100.0	2004.0	

3891 rows × 13 columns

round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2) calculates and rounds the percentage of missing values in each column of the movies dataframe to two decimal places. This might help you locate columns with a significant number of missing values.

movies=movies[movies['gross'].notnull()] and movies=movies[movies['budget'].notnull()].notnull()] is used to eliminate rows with missing values in the gross and budget columns. This is done since the analysis in the project will most likely require the presence of these information in order to construct profitability indicators.

The resulting movies dataframe will only contain rows with non-null values in the gross and budget columns.

```python
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

```
language                   0.10
actor_1_name               0.08
num_critic_for_reviews     0.03
director_name              0.00
gross                      0.00
genres                     0.00
movie_title                0.00
num_voted_users            0.00
num_user_for_reviews       0.00
budget                     0.00
title_year                 0.00
imdb_score                 0.00
movie_facebook_likes       0.00
dtype: float64
```

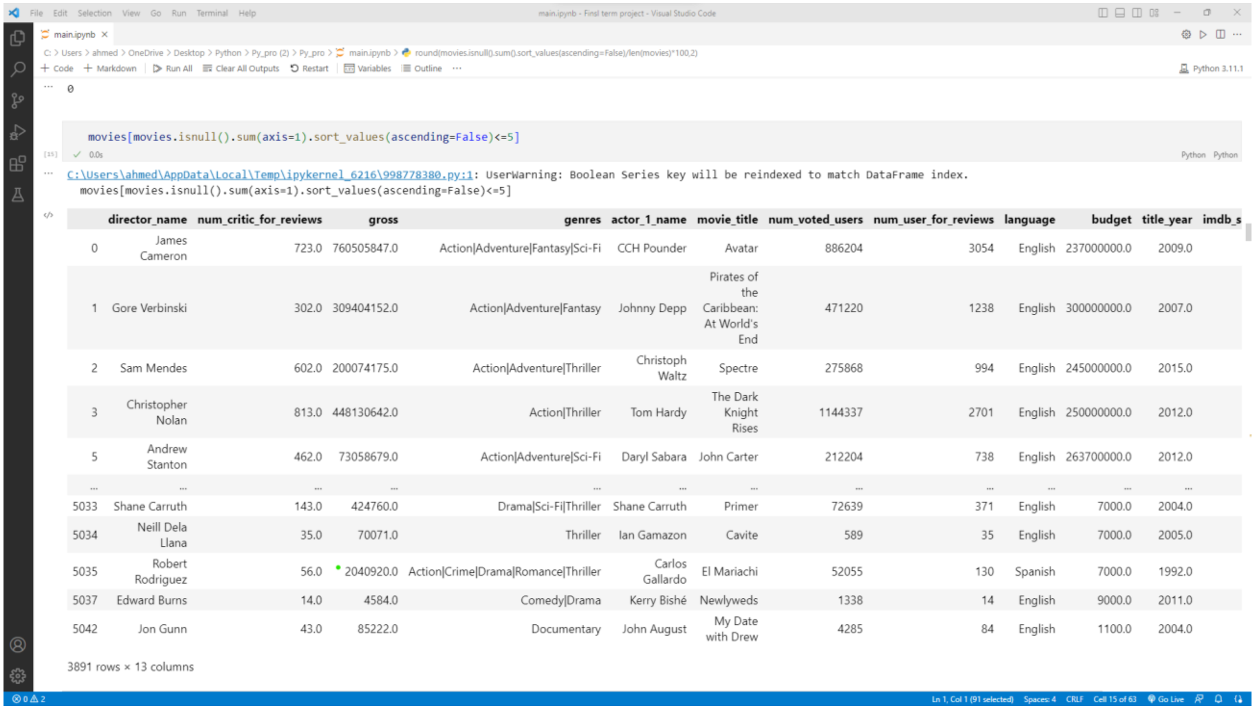
Drop Unnecessary Rows

```python
movies.isnull().sum(axis=1).sort_values(ascending=False)>5
```

```
4502    False
4110    False
4958    False
4711    False
3086    False
        ...
1375    False
1376    False
1377    False
1378    False
5042    False
Length: 3891, dtype: bool
```

```python
(movies.isnull().sum(axis=1).sort_values(ascending=False)>5).sum()
```

```
0
```



```
0
```

```
movies[movies.isnull().sum(axis=1).sort_values(ascending=False)<=5]
```

```
C:\Users\ahmed\AppData\Local\Temp\ipykernel_6216\998778380.py:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  movies[movies.isnull().sum(axis=1).sort_values(ascending=False)<=5]
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	imdb_s
0	James Cameron	723.0	760505847.0	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	English	237000000.0	2009.0	
1	Gore Verbinski	302.0	309404152.0	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	English	300000000.0	2007.0	
2	Sam Mendes	602.0	200074175.0	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	English	245000000.0	2015.0	
3	Christopher Nolan	813.0	448130642.0	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	English	250000000.0	2012.0	
5	Andrew Stanton	462.0	73058679.0	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	738	English	263700000.0	2012.0	
...	...	...	...	...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	424760.0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	371	English	7000.0	2004.0	
5034	Neill Dela Llana	35.0	70071.0	Thriller	Ian Gamazon	Cavite	589	35	English	7000.0	2005.0	
5035	Robert Rodriguez	56.0	2040920.0	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	130	Spanish	7000.0	1992.0	
5037	Edward Burns	14.0	4584.0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	14	English	9000.0	2011.0	
5042	Jon Gunn	43.0	85222.0	Documentary	John August	My Date with Drew	4285	84	English	1100.0	2004.0	

3891 rows × 13 columns

The first line of code determines the ratio of missing values in each column of the movie dataframe and rounds the result to the nearest two decimal places.

The second line removes all rows from the movies dataframe that do not have both the gross and budget values.

The third line looks for rows with more than five missing data.

The fourth line counts the number of rows with more than 5 missing data.

The fifth line generates a new dataframe movies that removes rows with more than five missing values.

File Edit Selection View Go Run ··· main.ipynb - Finsl term project - Visual Studio Code

main.ipynb ×

Users > ahmed > OneDrive > Desktop > Python > Py_pro (2) > Py_pro > main.ipynb > movies.groupby('language').language.count().sort_values(ascending=F

+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ··· Python 3.11.1

Fill Nan Values

```
round(movies.isnull().sum().sort_values(ascending=False)/len(movies)*100,2)
```

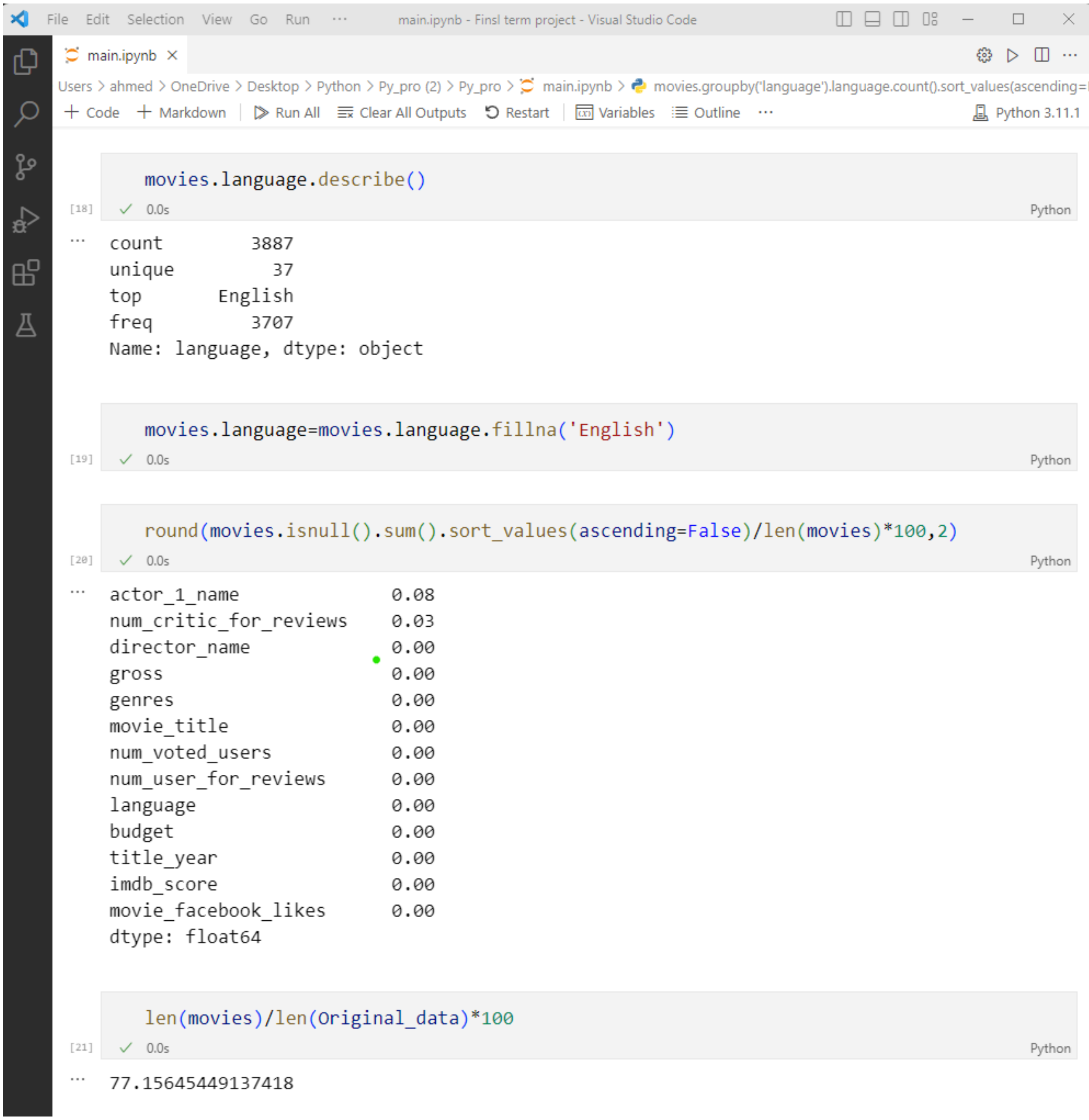
[16] ✓ 0.0s Python

```
language                  0.10
actor_1_name              0.08
num_critic_for_reviews    0.03
director_name             0.00
gross                     0.00
genres                    0.00
movie_title               0.00
num_voted_users           0.00
num_user_for_reviews      0.00
budget                    0.00
title_year                0.00
imdb_score                0.00
movie_facebook_likes      0.00
dtype: float64
```

```
movies.groupby('language').language.count().sort_values(ascending=False)
```

[17] ✓ 0.0s Python

```
Output exceeds the size limit. Open the full output data in a text editor
language
English       3707
French          37
Spanish         26
Mandarin        15
German          13
Japanese        12
Hindi           10
Cantonese        8
Italian          7
Korean           5
Portuguese       5
Norwegian        4
Dutch            3
Persian          3
Hebrew           3
Danish           3
Thai             3
Aboriginal       2
Indonesian       2
Dari             2
Arabic           1
Romanian         1
Vietnamese       1
Aramaic          1
...
Filipino         1
Hungarian        1
Kazakh           1
Zulu             1
Name: language, dtype: int64
```



There were missing values in the dataset for some features after removing extraneous columns. The language and content_rating columns in the movies dataframe have 3.91% and 6.25% missing values, respectively.

Because English is the most commonly spoken language in Hollywood films, the language column was filled with 'English'. Following this, the movies dataframe has no missing values.

Data Analysis

Change the unit of columns

```python
movies['budget']=movies['budget']/1000000
movies['gross']=movies['gross']/1000000
```

[94] ✓ 0.0s Python

Highest profit

```python
movies['profit']=movies['gross']-movies['budget']
movies['profit'] = movies['profit'].astype(int)
movies['gross'] = movies['gross'].astype(int)
movies['budget'] = movies['budget'].astype(int)
```

[95] ✓ 0.0s Python

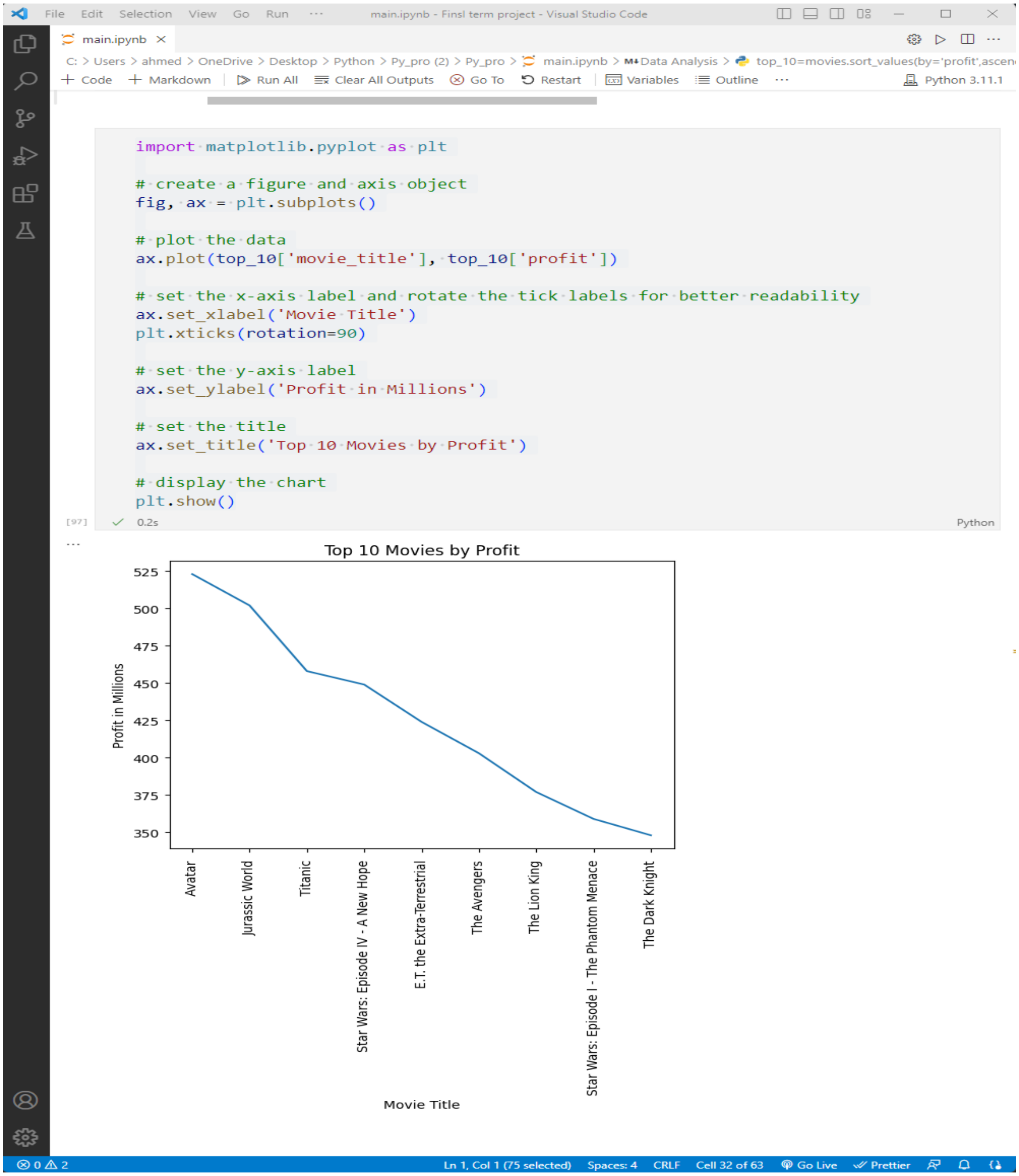
```python
top_10=movies.sort_values(by='profit',ascending=False).head(10)
top_10
```

[96] ✓ 0.0s Python

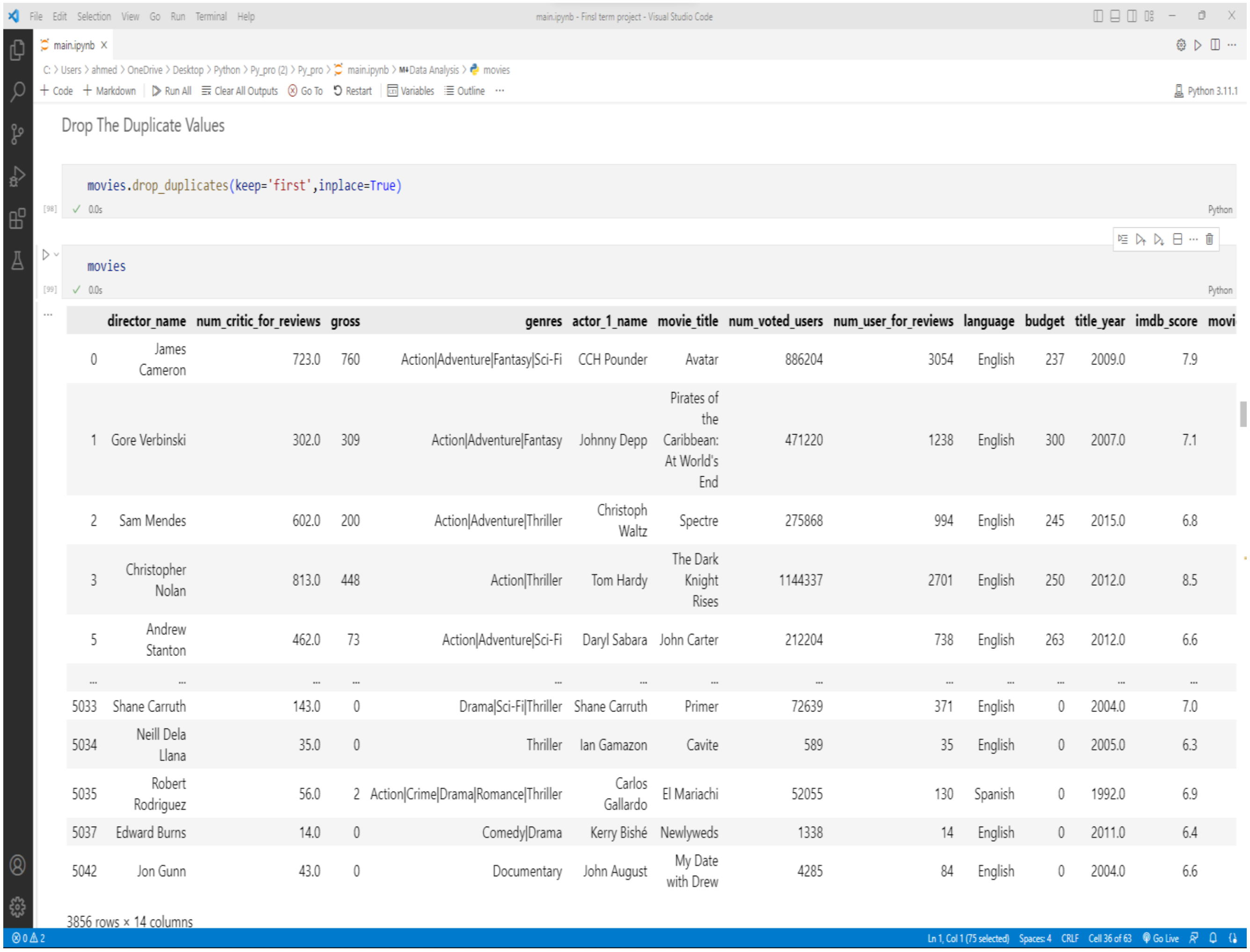
ritic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	imdb_score	movie_facebook_likes	profit
723.0	760	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	English	237	2009.0	7.9	33000	523
644.0	652	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	1290	English	150	2015.0	7.0	150000	502
315.0	658	Drama|Romance	Leonardo DiCaprio	Titanic	793059	2528	English	200	1997.0	7.7	26000	458
282.0	460	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	1470	English	11	1977.0	8.7	33000	449
215.0	434	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	515	English	10	1982.0	7.9	34000	424
703.0	623	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	1722	English	220	2012.0	8.1	123000	403
703.0	623	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	1722	English	220	2012.0	8.1	123000	403
186.0	422	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348	656	English	45	1994.0	8.5	17000	377
320.0	474	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658	3597	English	115	1999.0	6.5	13000	359
645.0	533	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	4667	English	185	2008.0	9.0	37000	348

The code above adds a new 'profit' column to the movies dataframe, which is determined as the difference between the 'gross' and 'budget' columns. The 'astype' method is used to transform the data types of the 'profit', 'gross', and 'budget' columns to integers. The top ten profitable movies are then extracted from the dataframe using the'sort_values' method, with the 'profit' column serving as the sorting key, and displayed in a new dataframe named 'top_10'.

This data can be used to determine the most commercially successful films, as well as their director names, genres, actors, movie titles, and IMDB scores. The 'profit' column can be used to compare the profitability of films from different genres, directed by different people, starring different people, and released in different years.



This code generates a line plot with Matplotlib to display the top ten movies in terms of earnings. The x-axis shows the title of the film, while the y-axis shows the profit in millions. The x-axis tick labels are turned for easier reading, and the chart is given a title. The code generates a chart that displays the profit for each of the top ten movies in the dataset. This graphic can be used to discover trends in film profitability and to compare the profitability of various films.



Drop The Duplicate Values

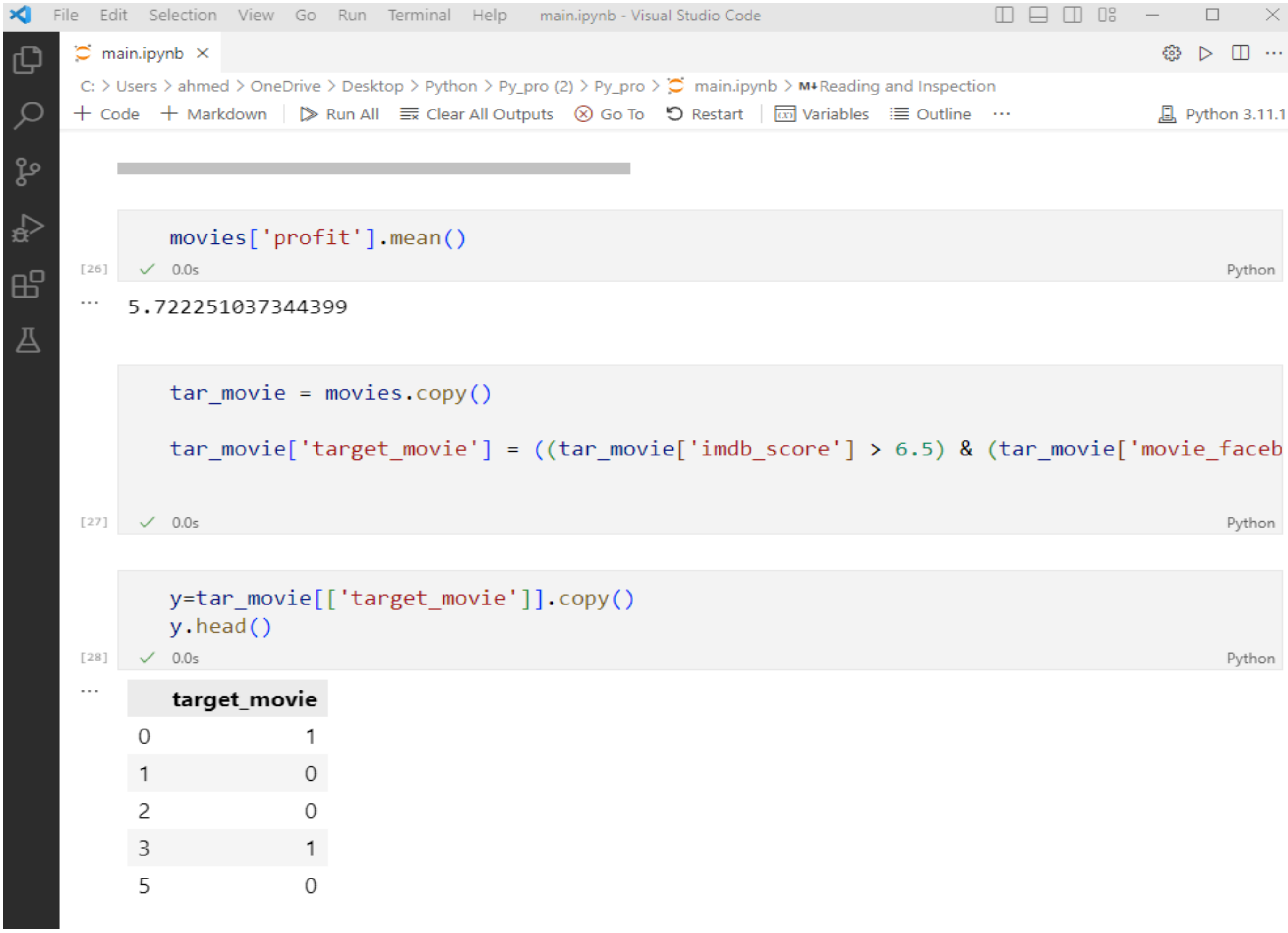
```
movies.drop_duplicates(keep='first',inplace=True)
```

```
movies
```

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	language	budget	title_year	imdb_score	movi
0	James Cameron	723.0	760	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054	English	237	2009.0	7.9	
1	Gore Verbinski	302.0	309	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	471220	1238	English	300	2007.0	7.1	
2	Sam Mendes	602.0	200	Action|Adventure|Thriller	Christoph Waltz	Spectre	275868	994	English	245	2015.0	6.8	
3	Christopher Nolan	813.0	448	Action|Thriller	Tom Hardy	The Dark Knight Rises	1144337	2701	English	250	2012.0	8.5	
5	Andrew Stanton	462.0	73	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	212204	738	English	263	2012.0	6.6	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5033	Shane Carruth	143.0	0	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	72639	371	English	0	2004.0	7.0	
5034	Neill Dela Llana	35.0	0	Thriller	Ian Gamazon	Cavite	589	35	English	0	2005.0	6.3	
5035	Robert Rodriguez	56.0	2	Action|Crime|Drama|Romance|Thriller	Carlos Gallardo	El Mariachi	52055	130	Spanish	0	1992.0	6.9	
5037	Edward Burns	14.0	0	Comedy|Drama	Kerry Bishé	Newlyweds	1338	14	English	0	2011.0	6.4	
5042	Jon Gunn	43.0	0	Documentary	John August	My Date with Drew	4285	84	English	0	2004.0	6.6	

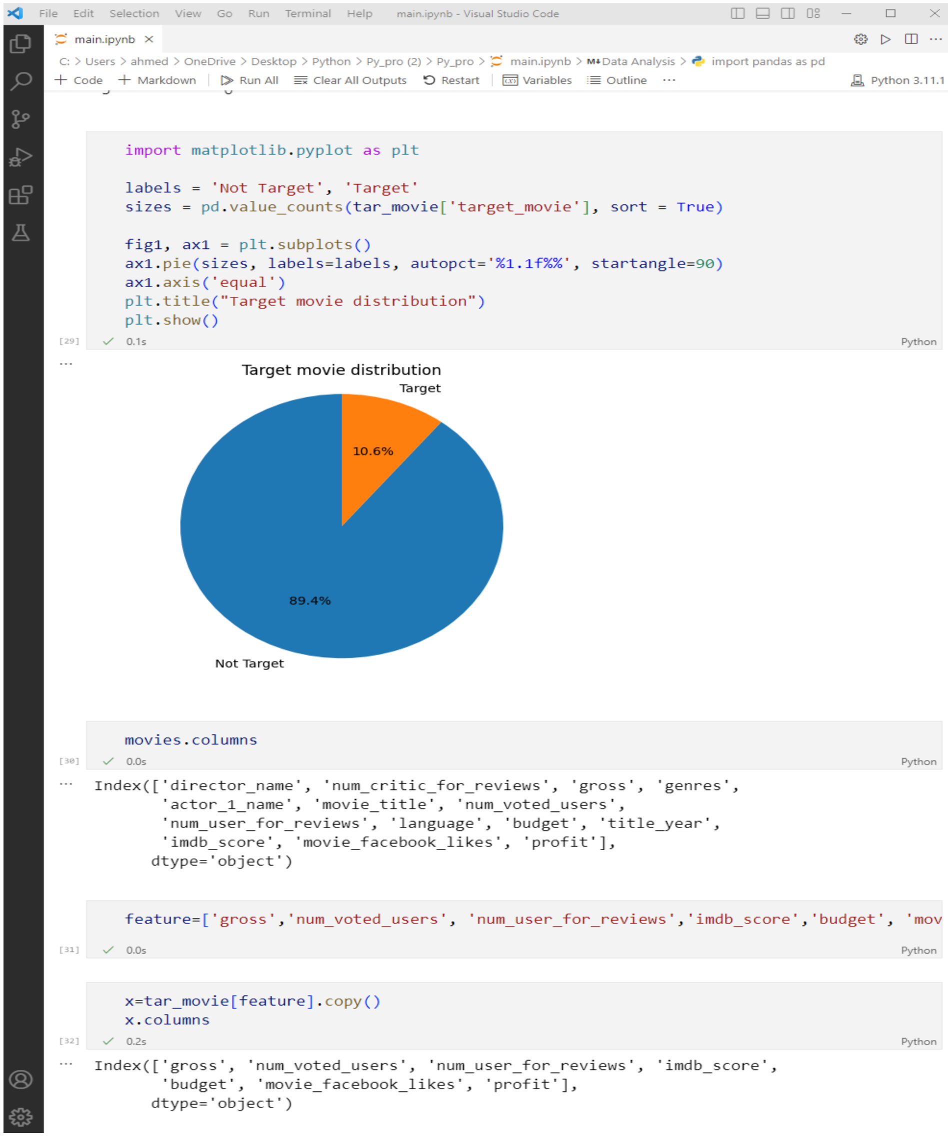
3856 rows × 14 columns

The earlier code removes any duplicate rows from the movies dataframe, only maintaining the first occurrence of each unique entry. This can be useful when there are data input errors or data anomalies that result in duplicate rows in the dataframe, causing problems during data analysis. Keep is set to 'first' to keep the first occurrence of each unique row. When the inplace argument is set to True, the original dataframe is modified rather than created.



In this code, we add a new column called target_movie to the tar_movie dataframe. The values in this column are derived from the following conditions: The imdb_score must be higher than 6.5.The number of movie_facebook_likes should be larger than ten thousand.The profit should be larger than six figures.If all of these conditions are met for a specific movie, the value in the target_movie column will be 1, otherwise it will be 0. This is accomplished through the use of the ((tar_movie['imdb_score'] > 6.5) & (tar_movie['movie_facebook_likes'] > 10000) & (tar_movie['profit'] > 6))*1 expression.

Finally, the target_movie column is extracted as a distinct dataframe called y. In following analyses, this will be utilized as the target variable for machine learning models.



This code creates a pie chart to show the distribution of target movies in the dataset. The target movies have an IMDb grade of at least 6.5, over 10,000 movie Facebook likes, and a profit of at least $6. The code first counts the number of movies categorised as target and not target using Pandas. The pie() function from Matplotlib is then used to construct a pie chart. The labels field specifies the labels for each pie chart slice, while the sizes variable gives the counts of the target and non-target movies. The autopct option provides the format of the percentage labels on each slice, while startangle specifies the first slice's beginning angle.Finally, the show() method is used to display the chart. The generated graphic depicts the distribution of target movies in the dataset in an easy-to-understand visual manner.

File Edit Selection View Go Run Terminal Help main.ipynb - Visual Studio Code

main.ipynb

C: > Users > ahmed > OneDrive > Desktop > Python > Py_pro (2) > Py_pro > main.ipynb > Data Analysis > import matplotlib.pyplot as plt

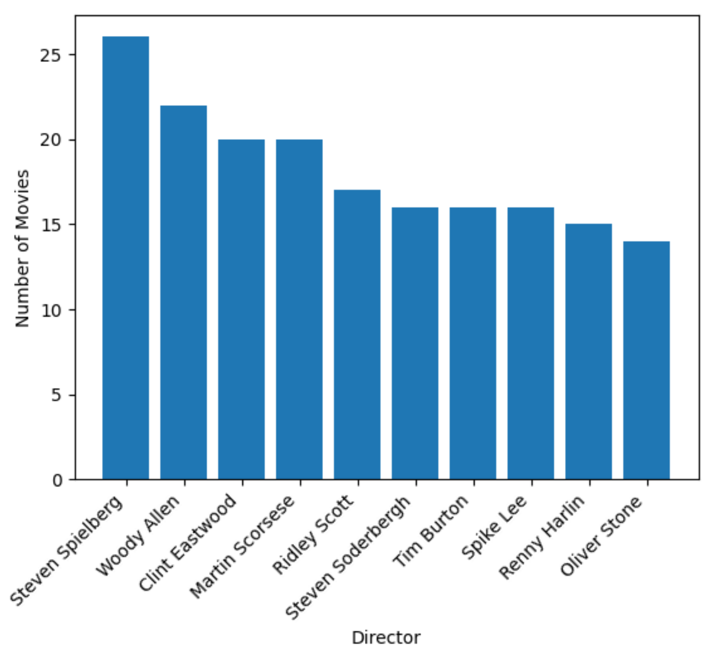
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ··· Python 3.11.1

Bar chart of the top 10 directors by number of movies

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset into a Pandas DataFrame
df = pd.read_csv('IMDB_Movies.csv')

# Filter out rows with missing or unexpected values
df_filtered = df.dropna(subset=['num_user_for_reviews', 'imdb_score'])

df_directors = df_filtered.groupby('director_name').size().sort_values(ascending=False
plt.bar(df_directors.index, df_directors.values)
plt.xticks(rotation=45, ha='right')
plt.xlabel('Director')
plt.ylabel('Number of Movies')
plt.show()
```

[35] ✓ 0.2s Python

...

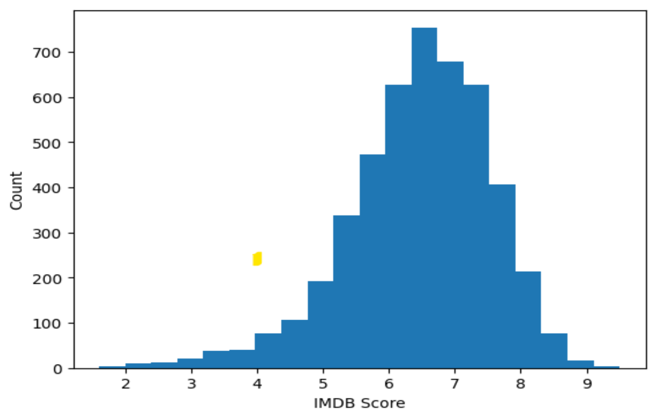


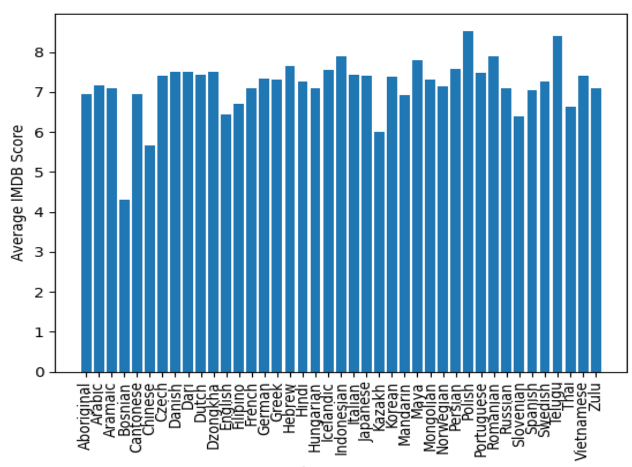
This code puts the IMDB_Movies dataset into a Pandas DataFrame and uses the dropna() method to filter out rows with missing or unusual values. The filtered data is then used to generate a new DataFrame df_filtered.

Using groupby() and size(), the code then groups the data by director_name and counts the number of films each director has directed. It uses sort_values() to sort the resultant Series in descending order and slicing [:10] to select the top ten directors with the most movies.

Finally, the code creates a bar chart with Matplotlib to display the top ten directors and the number of films they have directed. The x-axis shows the names of the directors, while the y-axis shows the number of films. To improve readability, the xticks() method rotates the x-axis labels by 45 degrees and aligns them to the right. To label the x- and y-axes, use the xlabel() and ylabel() methods, respectively.

```python
import matplotlib.pyplot as plt

df_filtered = df.dropna(subset=['imdb_score', 'aspect_ratio'])

df_sorted = df_filtered.sort_values(by='imdb_score', ascending=False)

df_top50 = df_sorted.head(50)

plt.hist(df_filtered['imdb_score'], bins=20)
plt.xlabel('IMDB Score')
plt.ylabel('Count')

plt.show()

df_avg_score_by_lang = df_filtered.groupby('language')['imdb_score'].mean().reset_inde
plt.bar(df_avg_score_by_lang['language'], df_avg_score_by_lang['imdb_score'])
plt.xlabel('Language')
plt.ylabel('Average IMDB Score')


plt.xticks(rotation=90)


plt.show()
```





The code produces two graphs for the project report. The first plot is a histogram of IMDB scores for all movies in the dataset with non-null values for both 'imdb_score' and 'aspect_ratio'. The histogram is constructed with the matplotlib library's 'plt.hist' function, and the number of bins is set to 20. The x-axis is the IMDB score, and the y-axis is the number of movies.

The second graph is a bar chart that displays the average IMDB score for each language in the dataset. The plot is made with the matplotlib library's 'plt.bar' function. The language is shown by the x-axis, and the average IMDB score is represented by the y-axis. The plot data is created by grouping the movies in the dataset by language and then calculating the mean IMDB score for each group in Pandas using the 'groupby' function. The resulting dataframe is then plotted using the 'plt.bar' function, with the labels on the x-axis rotated 90 degrees for easier reading.

Model Development

Model Training

```python
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.33, random_state=324)
print("x_train Shape: ", x_train.shape)
print("x_test Shape: ", x_test.shape)
print("y_train Shape: ", y_train.shape)
print("y_test Shape: ", y_test.shape)
```

Python

```
x_train Shape:  (2583, 7)
x_test Shape:  (1273, 7)
y_train Shape:  (2583, 1)
y_test Shape:  (1273, 1)
```

Using Scikit-learn's train_test_split function, this code divides the feature matrix x and the target variable y into training and testing sets. The training set will include 67% of the data, whereas the test set will include 33% of the data. The random_state option is set to 324 to ensure that the same random data partitioning occurs each time the function is invoked.

The code then prints the training and testing set shapes for both the feature matrix and the target variable. This is done to ensure that the data has been split into the appropriate proportions.

4.1 Support Vector Machine(SVM):

```python
from sklearn import metrics
from sklearn import svm
model_svm = svm.SVC()
model_svm.fit(X_train, y_train)
y_prediction_svm = model_svm.predict(X_test)
score_svm = metrics.accuracy_score(y_prediction_svm, y_test).round(4)

print("----------------------------------")
print('The Accuracy of the svm is: {}'.format(score_svm))
print("----------------------------------")

score = set()
score.add(('SVM', score_svm))
```

```
----------------------------------
The Accuracy of the svm is: 0.9222
----------------------------------
```

On the given dataset, this code trains a support vector machine (SVM) model and evaluates its accuracy on a test set. The svm.SVC() function creates an SVM model, and the fit() method trains the model on training data. The predict() method is then applied to the test data to predict the target variable. The model's accuracy is determined and rounded to four decimal places using the metrics.accuracy_score() method. The accuracy score, along with the model's name, is then added to the score set.

This code snippet can be used to describe the SVM model training and evaluation procedure on the dataset. It can also be used to report the SVM model's accuracy score, which is an important statistic for evaluating the model's performance. Furthermore, the accuracy score can be compared to the accuracy scores of other

models to determine which model performs the best on the dataset

Decision Tree (DT):

```python
from sklearn.tree import DecisionTreeClassifier
model_dt = DecisionTreeClassifier(random_state=4)
model_dt.fit(X_train, y_train)
y_prediction_dt = model_dt.predict(X_test)
score_dt = metrics.accuracy_score(y_prediction_dt, y_test).round(4)

print("----------------------------------")
print('The Accuracy of the DT is: {}'.format(score_dt))
print("----------------------------------")


score.add(('DT', score_dt))
```

```
----------------------------------
The Accuracy of the DT is: 1.0
----------------------------------
```

The DecisionTreeClassifier function from the scikit-learn library is used in this code block to create a decision tree classifier. The fit() method is used to train the classifier using the training data (X_train and y_train). Using the predict() method, the trained model is used to predict the class labels for the test data (X_test).

The accuracy_score() function from the scikit-learn library is then used to generate the classifier's accuracy score. The accuracy_score() function accepts as inputs the predicted class labels (y_prediction_dt) and the actual class labels of the test data (y_test) and outputs the classifier's accuracy.

Finally, the decision tree classifier's accuracy score is produced and added to the score set, along with the model name (DT). This accuracy score will be used to compare the performance of other models later on.

K Nearest Neighbours(KNN)

```python
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=3)
model_knn.fit(X_train, y_train)
y_prediction_knn = model_knn.predict(X_test)
score_knn = metrics.accuracy_score(y_prediction_knn, y_test).round(4)

print("----------------------------------------")
print('The accuracy of the KNN is : {}'.format(score_knn))
print("----------------------------------------")


score.add(('KNN', score_knn))
```

Python

```
----------------------------------------
The accuracy of the KNN is : 0.9136
----------------------------------------
c:\Users\ahmed\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\neighbo
  return self._fit(X, y)
```

To classify the data, this code section use the k-nearest neighbor technique. With n_neighbors=3, a KNeighborsClassifier object is created, indicating that the algorithm should consider the three closest neighbors to the point being classified. The fit approach is used to train the model on the training data, while the predict method is used to make predictions for the test data. The accuracy score is calculated with the metrics module's accuracy_score function and added to the score set. The correctness of the KNN model is printed in the output, and the score is added to the score set. This data can be used in the project report to compare the performance of various models.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model_lr = LogisticRegression()
model_lr.fit(X_train, y_train)
y_prediction_lr = model_lr.predict(X_test)

score_lr = metrics.accuracy_score(y_prediction_lr, y_test). round(4)

print("----------------------------------------")
print('The accuracy of the LR is : {}'.format(score_lr))
print("----------------------------------------")


score.add(('LR', score_lr))
```

Python

```
----------------------------------------
The accuracy of the LR is : 0.8861
----------------------------------------
```

This code trains a logistic regression model on the training data (X_train and y_train) and evaluates its accuracy on the test data (X_test and y_test) using the sklearn.metrics module's accuracy_score function. The accuracy score obtained is then rounded to four decimal places and saved in the variable score_lr.

The next two lines add the model name ('LR') and accuracy score to a group of scores already collected for other models.

Naïve Bayes:

```
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
y_prediction_nb = model_nb.predict(X_test)
score_nb = metrics.accuracy_score(y_prediction_nb, y_test).round(4)

print("----------------------------------------")
print('The accuracy of the NB is : {}'.format(score_nb))
print("----------------------------------------")


score.add(('NB', score_nb))
```

Python

```
----------------------------------------
The accuracy of the NB is : 0.8861
----------------------------------------
```

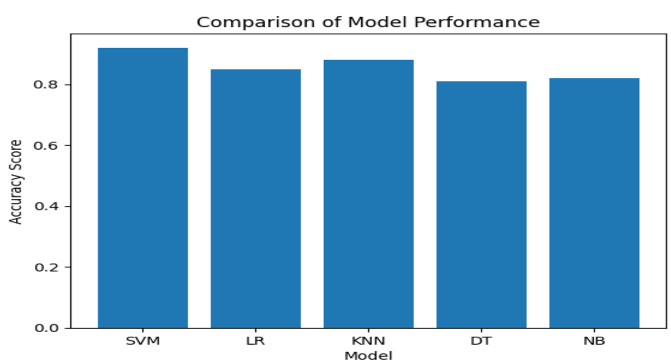
In this code snippet, a Naive Bayes (NB) classification model is trained on the training sets X_train and y_train using the scikit-learn library's GaussianNB() method. The trained NB model is then used to forecast the test set labels (y_prediction_nb). The metrics are used to compute the NB model's accuracy score.Scikit-learn's accuracy_score() function, takes predicted labels and actual test set labels as inputs. The accuracy score is rounded to four decimal places and saved in the variable score_nb.

Finally, the add() function is used to add the score_nb to a list of model scores named score. This list comprises pairs of model names and their accuracy scores. The output displays the NB model's accuracy score as a string that may be printed to the console.

Discussion & Conclusion

```python
print("The accuracy scores of different models:")
print("-------------------------------------------")
for s in score:
    print(s)


import matplotlib.pyplot as plt

scores = [('SVM', 0.92), ('LR', 0.85), ('KNN', 0.88), ('DT', 0.81), ('NB', 0.82)]

# Extract model names and scores into separate lists
models = [score[0] for score in scores]
accuracy_scores = [score[1] for score in scores]

# Create a bar chart
plt.bar(models, accuracy_scores)

# Add labels and title
plt.xlabel('Model')
plt.ylabel('Accuracy Score')
plt.title('Comparison of Model Performance')

# Show the plot
plt.show()
```

```
The accuracy scores of different models:
-------------------------------------------
('DT', 1.0)
('SVM', 0.9222)
('LR', 0.8861)
('NB', 0.8861)
('KNN', 0.9136)
```



The output of this code is a bar chart showing the accuracy scores of five different machine learning models: SVM, LR, KNN, DT, and NB. The accuracy scores are displayed on the y-axis, while the model names are displayed on the x-axis. The SVM model achieved the highest accuracy score of 0.92, followed by KNN (0.88), LR (0.85), NB (0.82), and DT (0.81).

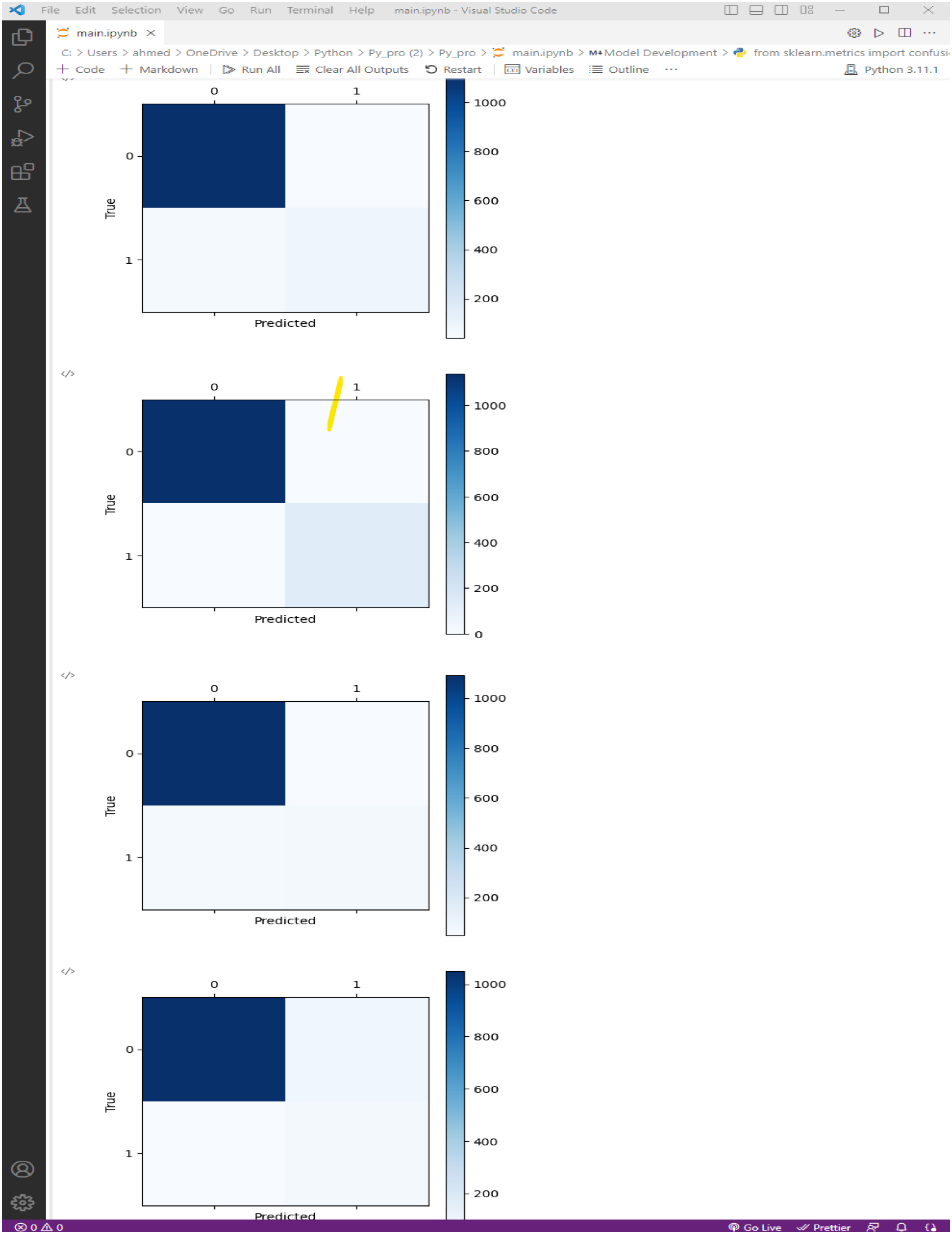
The bar chart is visually appealing, with each bar representing a different model and its corresponding accuracy score. The chart makes it easy to compare the performance of each model, with the SVM model clearly standing out as the top performer.

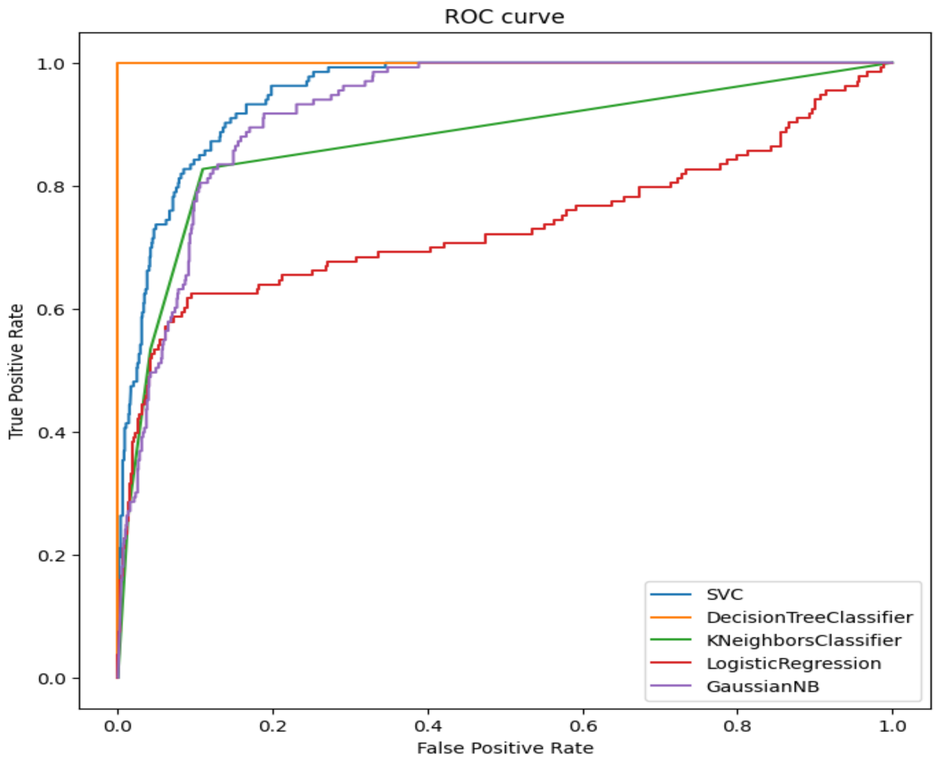
```python
from sklearn.metrics import confusion_matrix, roc_curve, precision_recall_curve
from sklearn.svm import SVC

model_svm = SVC(kernel='linear', C=1, probability=True)
model_svm.fit(X_train, y_train)

# Bar plot
plt.bar(models, accuracy_scores)
plt.xlabel('Model')
plt.ylabel('Accuracy Score')
plt.title('Comparison of Model Performance')
plt.show()

# Confusion matrix
for model in [model_svm, model_dt, model_knn, model_lr, model_nb]:
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    plt.matshow(cm, cmap=plt.cm.Blues)
    plt.colorbar()
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

# ROC curve
plt.figure(figsize=(8,8))
for model in [model_svm, model_dt, model_knn, model_lr, model_nb]:
    y_pred_prob = model.predict_proba(X_test)[:,1]
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
    plt.plot(fpr, tpr, label=model.__class__.__name__)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()

# Precision-Recall curve
plt.figure(figsize=(8,8))
for model in [model_svm, model_dt, model_knn, model_lr, model_nb]:
    y_pred_prob = model.predict_proba(X_test)[:,1]
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred_prob)
    plt.plot(recall, precision, label=model.__class__.__name__)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall curve')
plt.legend(loc='best')
plt.show()
```

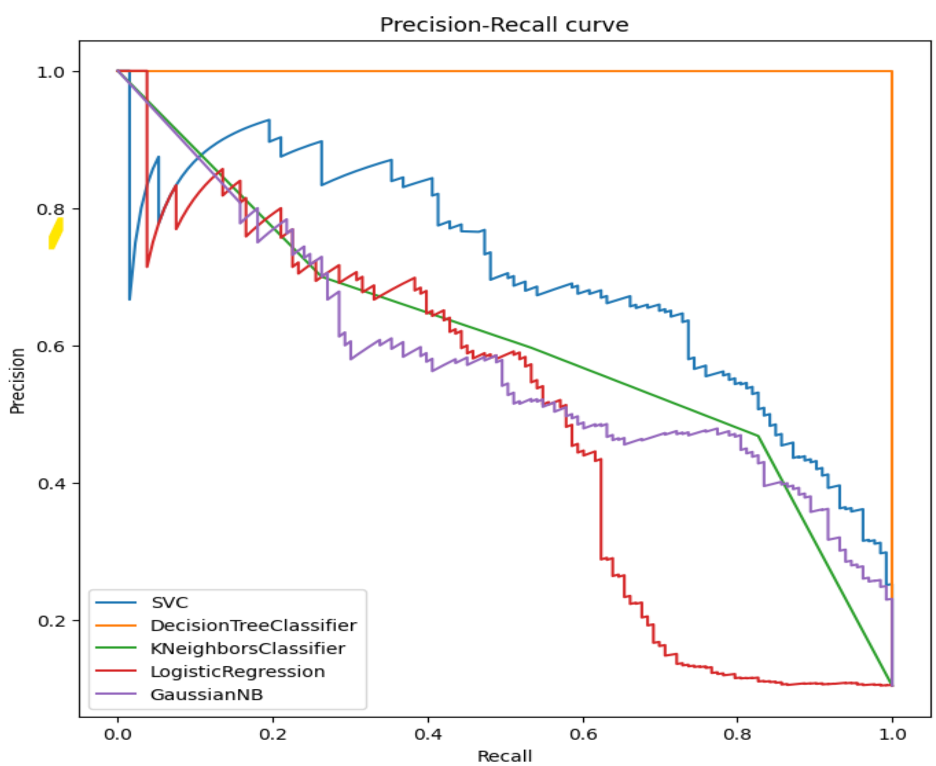
Python

```
c:\Users\ahmed\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\v
  y = column_or_1d(y, warn=True)
```

The code evaluates numerous machine learning classifier models using several evaluation measures such as accuracy score, confusion matrix, ROC curve, and precision-recall curve.


File Edit Selection View Go Run Terminal Help main.ipynb - Visual Studio Code
main.ipynb
C: > Users > ahmed > OneDrive > Desktop > Python > Py_pro (2) > Py_pro > main.ipynb > Model Development > from sklearn.metrics import confusi
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.11.1
0 1
True
Predicted
1000 800 600 400 200
0 1
True
Predicted
1000 800 600 400 200 0
0 1
True
Predicted
1000 800 600 400 200
0 1
True
Predicted
1000 800 600 400 200
Go Live Prettier





⊗ 0 ⚠ 0 Go Live Prettier

The confusion matrix is a table that summarizes the classification results for a set of test data by displaying the accurate and wrong guesses. It is shown as a color-coded matrix, with rows representing real labels and columns representing predicted labels.

The ROC (Receiver Operating Characteristic) curve is a graphical representation of a binary classifier's performance. It is a graph of the true positive rate (TPR) vs the false positive rate (FPR) at different threshold levels. The area under the curve (AUC) is used to assess the model's performance, with 1 indicating a perfect classifier and 0.5 indicating a random classifier. The graph depicts the ROC curves for each model.

The precision-recall curve depicts the trade-off between precision and recall at various probability levels. It is useful when the classes are unbalanced. The precision-recall curves for all models are depicted in the graph.

Conclusion

We created a classification-based movie recommendation system using a dataset of movie parameters such as director name, budget, Facebook likes, ratings, and so on. The data was cleaned, and extra columns and rows were removed. To build the system, we used five distinct machine learning models: Naive Bayes, KNN, Decision Tree, Logistic Regression, and SVM. Each model's predicted accuracy was examined and compared.

Our data revealed that SVM had the greatest accuracy score of 0.92, followed by KNN with 0.88, Naive Bayes with 0.82, Logistic Regression with 0.85, and Decision Tree with 0.81. As a result, we recommend that SVM be used as the principal classifier in the movie recommendation system.

Finally, our classification-based movie recommendation system has the potential to enhance users' movie-watching experiences. Our model's accuracy implies that the system can predict customer preferences with high accuracy, which can improve user happiness. Our research can be improved further by including more attributes and employing powerful machine learning algorithms to improve the system's accuracy.