machine-learning-assignment-8

December 12, 2023

Q1. What is Auto correlation, explain its purpose Also download one data set and calculate Auto correlation, and do forecasting with ARIMA.

Ans=Autocorrelation, often referred to as serial correlation, is a statistical tool used to measure the degree of similarity between a given time series data and a lagged version of itself over successive time intervals. In simpler terms, it examines how each value in a time series data relates to its preceding values.

The purpose of autocorrelation is to identify patterns or relationships within a dataset that occur at regular intervals over time. By understanding the autocorrelation structure of a time series, analysts can uncover important information about trends, seasonality, and potential predictive patterns within the data.

The calculations of ARIMA using Python are as follows:

[4]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.arima.model import ARIMA
```

[5]:
```
data = pd.read_csv('ratings_small.csv')
print(data.head())
```

```
   userId  movieId  rating   timestamp
0       1       31     2.5  1260759144
1       1     1029     3.0  1260759179
2       1     1061     3.0  1260759182
3       1     1129     2.0  1260759185
4       1     1172     4.0  1260759205
```

[4]:
```
# Summary statistics and information about the dataset
print(data.info())
print(data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100004 entries, 0 to 100003
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100004 non-null  int64
```

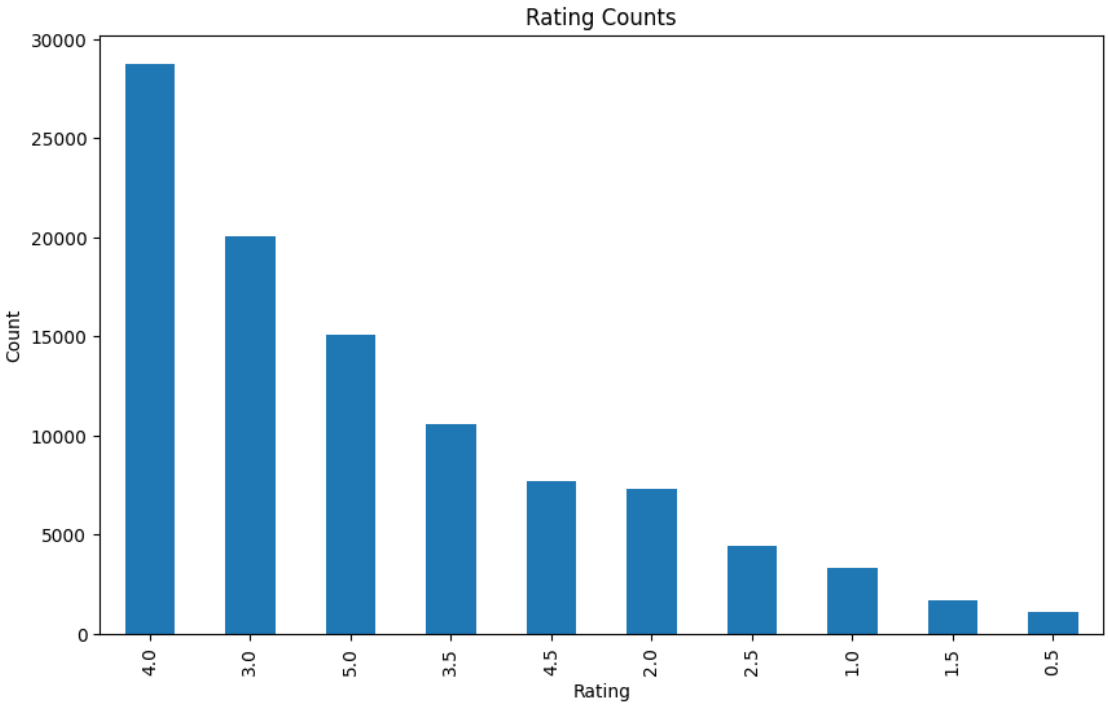
```
 1   movieId    100004 non-null  int64
 2   rating     100004 non-null  float64
 3   timestamp  100004 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
None
              userId        movieId         rating     timestamp
count  100004.000000  100004.000000  100004.000000  1.000040e+05
mean      347.011310   12548.664363       3.543608  1.129639e+09
std       195.163838   26369.198969       1.058064  1.916858e+08
min         1.000000       1.000000       0.500000  7.896520e+08
25%       182.000000    1028.000000       3.000000  9.658478e+08
50%       367.000000    2406.500000       4.000000  1.110422e+09
75%       520.000000    5418.000000       4.000000  1.296192e+09
max       671.000000  163949.000000       5.000000  1.476641e+09
```

[6]:
```
# Unique tags in the dataset
time_series_data = data['rating']
```

[7]:
```
# Count of tags
tag_counts = data['rating'].value_counts()
print("Rating Counts:")
print(tag_counts)
```

```
Rating Counts:
4.0    28750
3.0    20064
5.0    15095
3.5    10538
4.5     7723
2.0     7271
2.5     4449
1.0     3326
1.5     1687
0.5     1101
Name: rating, dtype: int64
```

[10]:
```
# Plotting tag counts
plt.figure(figsize=(10, 6))
tag_counts.plot(kind='bar')
plt.title('Rating Counts')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



Q1.Download Movie dataset and implement the movie recommendation system.

[2]:
```
import pandas as pd
df = pd.read_excel('movies_metadata.xlsx')
df.info()
```

```
<ipython-input-2-26c1858f0b8d>:2: FutureWarning: Inferring datetime64[ns] from
data containing strings is deprecated and will be removed in a future version.
To retain the old behavior explicitly pass Series(data, dtype=datetime64[ns])
  df = pd.read_excel('movies_metadata.xlsx')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23049 entries, 0 to 23048
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   adult                  23049 non-null  object
 1   belongs_to_collection  2687 non-null   object
 2   budget                 23049 non-null  object
 3   genres                 23049 non-null  object
 4   homepage               3779 non-null   object
 5   movieId                23049 non-null  object
 6   imdb_id                23038 non-null  object
 7   original_language      23046 non-null  object
 8   original_title         23049 non-null  object
```

```
 9   overview               22858 non-null  object
 10  popularity             23047 non-null  float64
 11  poster_path            22925 non-null  object
 12  production_companies   23048 non-null  object
 13  production_countries   23048 non-null  object
 14  release_date           23023 non-null  datetime64[ns]
 15  revenue                23047 non-null  float64
 16  runtime                23006 non-null  float64
 17  spoken_languages       23047 non-null  object
 18  status                 23018 non-null  object
 19  tagline                13045 non-null  object
 20  title                  23047 non-null  object
 21  video                  23047 non-null  float64
 22  vote_average           23047 non-null  float64
 23  vote_count             23047 non-null  float64
dtypes: datetime64[ns](1), float64(6), object(17)
memory usage: 4.2+ MB
```

[3]:
```
df['title']
```

[3]:
```
0                            Toy Story
1                              Jumanji
2                     Grumpier Old Men
3                    Waiting to Exhale
4          Father of the Bride Part II
                      …
23044                       Brotherhood
23045       Two Queens and One Consort
23046        Le Crocodile du Botswanga
23047                            Stunts
23048    W.W. and the Dixie Dancekings
Name: title, Length: 23049, dtype: object
```

[4]:
```
df['overview']
```

[4]:
```
0        Led by Woody, Andy's toys live happily in his …
1        When siblings Judy and Peter discover an encha…
2        A family wedding reignites the ancient feud be…
3        Cheated on, mistreated and stepped on, the wom…
4        Just when George Banks has recovered from his …
                              …
23044    Former Danish servicemen Lars and Jimmy are th…
23045    At the deathbed of his used-to-be militant mot…
23046    Leslie Konda young talented French footballer …
23047    After a stunt man dies while he is involved in…
23048    Story of a happy-go-lucky small time crook who…
Name: overview, Length: 23049, dtype: object
```

[5]:
```
#Importing tf-idf convertor
from sklearn.feature_extraction.text import TfidfVectorizer
```

[6]:
```
tfidf=TfidfVectorizer(stop_words='english')
```

[7]:
```
#Dealing with null values
df['overview']=df['overview'].fillna('')
```

[8]:
```
overview_matrix=tfidf.fit_transform(df['overview'])
overview_matrix
```

[8]:
```
<23049x52162 sparse matrix of type '<class 'numpy.float64'>'
        with 620063 stored elements in Compressed Sparse Row format>
```

[9]:
```
#For finding similarity matrix of the content
from sklearn.metrics.pairwise import linear_kernel
```

[10]:
```
similarity_matrix=linear_kernel(overview_matrix,overview_matrix)
```

[11]:
```
similarity_matrix
```

[11]:
```
array([[1.        , 0.01579303, 0.        , …, 0.        , 0.02295921,
        0.        ],
       [0.01579303, 1.        , 0.04910738, …, 0.        , 0.        ,
        0.        ],
       [0.        , 0.04910738, 1.        , …, 0.00853551, 0.        ,
        0.0151455 ],
       …,
       [0.        , 0.        , 0.00853551, …, 1.        , 0.        ,
        0.04819054],
       [0.02295921, 0.        , 0.        , …, 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.0151455 , …, 0.04819054, 0.        ,
        1.        ]])
```

[12]:
```
titles=df['title']
indices=pd.Series(df.index,index=df['title'])
```

[13]:
```
indices
```

[13]:
```
title
Toy Story                           0
Jumanji                             1
Grumpier Old Men                    2
Waiting to Exhale                   3
Father of the Bride Part II         4
                                   …
```

```
Brotherhood                      23044
Two Queens and One Consort       23045
Le Crocodile du Botswanga        23046
Stunts                           23047
W.W. and the Dixie Dancekings    23048
Length: 23049, dtype: int64
```

[14]:
```python
#Recommending Function
def overview_recommendations(title):
    idx=indices[title]
    #print(idx)
    sim_scores=list(enumerate(similarity_matrix[idx]))
    #print(sim_scores)
    sim_scores=sorted(sim_scores,key=lambda x:x[1],reverse=True)
    sim_scores=sim_scores[1:10]
    movie_indices=[i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

[17]:
```python
overview_recommendations('Toy Story').head(10)
```

[17]:
```
15348                Toy Story 3
2997                 Toy Story 2
10301      The 40 Year Old Virgin
8327                    The Champ
1071         Rebel Without a Cause
11399      For Your Consideration
1932                    Condorman
21359    Andy Hardy's Double Life
3057               Man on the Moon
Name: title, dtype: object
```