INTERNSHIP PROJECT REPORT ON MACHINE LEARNING

SUBMITTED BY:

Bhaswati Deka(22BEC0325)

Marjana Bhuyan(23BCB0145)

Table of contents

Introduction

During our internship focused on machine learning, we delved into fundamental concepts that form the backbone of predictive analytics and data-driven decision-making. One of the key components we explored was the intricate world of datasets. Understanding datasets involves more than just gathering raw information; it entails comprehending the structure, cleaning, and preprocessing data to derive meaningful insights. The datasets we used were Iris,Forest Fire and Heart Disesase. Leveraging the powerful capabilities of libraries such as NumPy and Pandas within Jupyter Notebook, we learned how to manipulate datasets, extract essential parameters, and prepare data for analysis. These skills proved indispensable as we embarked on two distinct projects: predicting salaries through regression analysis and tackling spam emails using Naive Bayes classification.

Project I: Predicting Salaries Using Regression

In the realm of predictive modeling, regression analysis emerged as a pivotal tool for projecting salary outcomes based on various factors. Our project involved constructing regression models to discern patterns and relationships within a dataset that influences salary levels. Through careful feature selection, model training, and evaluation using techniques like linear regression and decision trees, we aimed to develop robust models capable of predicting salaries accurately. This endeavor not only sharpened my understanding of regression techniques but also honed my ability to interpret and communicate findings derived from statistical models.

Project II: Book Recommender System

During our internship, we had the opportunity to contribute to an exciting project called the Book Recommendation System using Machine Learning. This project aimed to leverage advanced algorithms to personalize book recommendations based on user preferences and behavior. By analyzing large datasets of book ratings and user interactions(dataset from Kaggle: source: kaggle datasets download -d arashnic/book-recommendation-dataset), we implemented collaborative filtering techniques to suggest books that users are likely to enjoy, thereby enhancing their reading experience. Throughout this project,we gained valuable insights into data preprocessing, algorithm selection, and model evaluation, which were crucial in refining the system's accuracy and usability. This experience not only deepened our understanding of machine learning applications but also equipped us with practical skills in handling real-world data and developing recommendation systems.. This project not only underscored the importance of feature engineering and model tuning but also highlighted the practical implications of machine learning in real-world scenarios.

Dataset 1: Iris

- Reading the dataset

jupyter task1 Last Checkpoint: 06/26/2024 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
!pip install numpy
```
```
Requirement already satisfied: numpy in c:\users\dekab\anaconda3\lib\site-packages (1.21.5)
```

In [2]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('iris.csv')
print(df)
```
```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
0              5.1           3.5            1.4           0.2     Iris-setosa
1              4.9           3.0            1.4           0.2     Iris-setosa
2              4.7           3.2            1.3           0.2     Iris-setosa
3              4.6           3.1            1.5           0.2     Iris-setosa
4              5.0           3.6            1.4           0.2     Iris-setosa
..             ...           ...            ...           ...             ...
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
```

- Dropping a column

In [3]:
```
df=df.drop(['Species'],axis=1)
print(df)
```
```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0              5.1           3.5            1.4           0.2
1              4.9           3.0            1.4           0.2
2              4.7           3.2            1.3           0.2
3              4.6           3.1            1.5           0.2
4              5.0           3.6            1.4           0.2
..             ...           ...            ...           ...
145            6.7           3.0            5.2           2.3
146            6.3           2.5            5.0           1.9
147            6.5           3.0            5.2           2.0
148            6.2           3.4            5.4           2.3
149            5.9           3.0            5.1           1.8

[150 rows x 4 columns]
```

- Converting the dataset into array

In [4]:
```
data = df.iloc[:,:4].values
data
```
Out[4]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

- Finding the mean

```
In [5]: df.mean(axis=1)

Out[5]: 0      2.550
        1      2.375
        2      2.350
        3      2.350
        4      2.550
               ...
        145    4.300
        146    3.925
        147    4.175
        148    4.325
        149    3.950
        Length: 150, dtype: float64
```

- Finding the mean using loop

```
In [8]: mean={}
        for column in df.columns:
            sumv=0
            for value in df[column]:
                sumv+=value
            mean[column]=sumv/len(df)

        print("Mean:")
        for feature, mean in mean.items():
            print(f"{feature}: {mean}")

        Mean:
        SepalLengthCm: 5.843333333333335
        SepalWidthCm: 3.0540000000000007
        PetalLengthCm: 3.7586666666666693
        PetalWidthCm: 1.1986666666666672
```

- Finding the Variance using loop

```
In [13]: var={}
         for column in df.columns:
             mean=meanf=a[:,0s[column]
             sumsqdif=0
             for value in df[column]:
                 sumsqdif+=(value-mean)**2
             var[column]=sumsqdif/(len(df)-1)

         print("Variance:")
         for feature, var in var.items():
             print(f"{feature}: {var}")
```

- Finding various parameters using *describe* keyword

```
In [18]: print('Describe')
         print(df.describe())

         Describe
                SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
         count     150.000000    150.000000     150.000000    150.000000
         mean        5.843333      3.054000       3.758667      1.198667
         std         0.828066      0.433594       1.764420      0.763161
         min         4.300000      2.000000       1.000000      0.100000
         25%         5.100000      2.800000       1.600000      0.300000
         50%         5.800000      3.000000       4.350000      1.300000
         75%         6.400000      3.300000       5.100000      1.800000
         max         7.900000      4.400000       6.900000      2.500000
```

- Applying Naïve Bayes Algorithm on Iris Dataset

NaiveBayes_Iris

In [1]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('Iris.csv')
df
```

Out[1]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [3]:
```
x=df.drop(["Species"],axis=1)
y=df["Species"]
```

In [9]:
```
from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()
x_scaled=scaler.fit_transform(x)
x_scaled
```

Out[9]:
```
array([[0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667],
       [0.30555556, 0.79166667, 0.11864407, 0.125     ],
       [0.08333333, 0.58333333, 0.06779661, 0.08333333],
       [0.19444444, 0.58333333, 0.08474576, 0.04166667],
       [0.02777778, 0.375     , 0.06779661, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.30555556, 0.70833333, 0.08474576, 0.04166667],
       [0.13888889, 0.58333333, 0.10169492, 0.04166667],
       [0.13888889, 0.41666667, 0.06779661, 0.        ],
       [0.        , 0.41666667, 0.01694915, 0.        ],
       [0.41666667, 0.83333333, 0.03389831, 0.04166667],
       [0.38888889, 1.        , 0.08474576, 0.125     ],
       [0.30555556, 0.79166667, 0.05084746, 0.125     ],
       [0.22222222, 0.625     , 0.06779661, 0.08333333],
       [0.38888889, 0.75      , 0.11864407, 0.08333333],
```

In [6]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
```

In [7]:
```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(x_train, y_train)

y_pred= gnb.predict(x_test)
```

In [8]:
```
y_pred
```

Out[8]:
```
73      Iris-versicolor
18          Iris-setosa
118      Iris-virginica
78      Iris-versicolor
76      Iris-versicolor
31          Iris-setosa
64      Iris-versicolor
141      Iris-virginica
68      Iris-versicolor
82      Iris-versicolor
110      Iris-virginica
12          Iris-setosa
36          Iris-setosa
9           Iris-setosa
19          Iris-setosa
56      Iris-versicolor
104      Iris-virginica
69      Iris-versicolor
55      Iris-versicolor
132      Iris-virginica
29          Iris-setosa
127      Iris-virginica
26          Iris-setosa
128      Iris-virginica
131      Iris-virginica
145      Iris-virginica
108      Iris-virginica
143      Iris-virginica
45          Iris-setosa
30          Iris-setosa
Name: Species, dtype: object
```

In [10]:
```
y_test
```

Out[10]:
```
73      Iris-versicolor
18          Iris-setosa
118      Iris-virginica
78      Iris-versicolor
76      Iris-versicolor
31          Iris-setosa
64      Iris-versicolor
141      Iris-virginica
68      Iris-versicolor
82      Iris-versicolor
110      Iris-virginica
12          Iris-setosa
36          Iris-setosa
9           Iris-setosa
19          Iris-setosa
56      Iris-versicolor
104      Iris-virginica
69      Iris-versicolor
55      Iris-versicolor
132      Iris-virginica
29          Iris-setosa
127      Iris-virginica
26          Iris-setosa
128      Iris-virginica
131      Iris-virginica
145      Iris-virginica
108      Iris-virginica
143      Iris-virginica
45          Iris-setosa
30          Iris-setosa
Name: Species, dtype: object
```

In [1]:
```
# predicted y values and test values are same
```

Dataset 2: Forest Fire

- ## Reading the data

In [1]:
```
import pandas as pd
df=pd.read_csv('forestfires.csv')
print(df)
```

```
       X  Y month  day  FFMC    DMC     DC   ISI  temp  RH  wind  rain   area
0      7  5   mar  fri  86.2   26.2   94.3   5.1   8.2  51   6.7   0.0   0.00
1      7  4   oct  tue  90.6   35.4  669.1   6.7  18.0  33   0.9   0.0   0.00
2      7  4   oct  sat  90.6   43.7  686.9   6.7  14.6  33   1.3   0.0   0.00
3      8  6   mar  fri  91.7   33.3   77.5   9.0   8.3  97   4.0   0.2   0.00
4      8  6   mar  sun  89.3   51.3  102.2   9.6  11.4  99   1.8   0.0   0.00
..    .. ..   ...  ...   ...    ...    ...   ...   ...  ..   ...   ...    ...
512    4  3   aug  sun  81.6   56.7  665.6   1.9  27.8  32   2.7   0.0   6.44
513    2  4   aug  sun  81.6   56.7  665.6   1.9  21.9  71   5.8   0.0  54.29
514    7  4   aug  sun  81.6   56.7  665.6   1.9  21.2  70   6.7   0.0  11.16
515    1  4   aug  sat  94.4  146.0  614.7  11.3  25.6  42   4.0   0.0   0.00
516    6  3   nov  tue  79.5    3.0  106.7   1.1  11.8  31   4.5   0.0   0.00

[517 rows x 13 columns]
```

In [3]:
```
df=df.drop(['month'],axis=1)
df=df.drop(['day'],axis=1)
print(df)
```

```
       X  Y  FFMC    DMC     DC   ISI  temp  RH  wind  rain   area
0      7  5  86.2   26.2   94.3   5.1   8.2  51   6.7   0.0   0.00
1      7  4  90.6   35.4  669.1   6.7  18.0  33   0.9   0.0   0.00
2      7  4  90.6   43.7  686.9   6.7  14.6  33   1.3   0.0   0.00
3      8  6  91.7   33.3   77.5   9.0   8.3  97   4.0   0.2   0.00
4      8  6  89.3   51.3  102.2   9.6  11.4  99   1.8   0.0   0.00
..    .. ..   ...    ...    ...   ...   ...  ..   ...   ...    ...
512    4  3  81.6   56.7  665.6   1.9  27.8  32   2.7   0.0   6.44
513    2  4  81.6   56.7  665.6   1.9  21.9  71   5.8   0.0  54.29
514    7  4  81.6   56.7  665.6   1.9  21.2  70   6.7   0.0  11.16
515    1  4  94.4  146.0  614.7  11.3  25.6  42   4.0   0.0   0.00
516    6  3  79.5    3.0  106.7   1.1  11.8  31   4.5   0.0   0.00

[517 rows x 11 columns]
```

- ## Finding the mean

In [4]:
```
mean={}
for column in df.columns:
    sumv=0
    for value in df[column]:
        sumv+=value
    mean[column]=sumv/len(df)

print("Mean:")
for feature, mean in mean.items():
    print(f"{feature}: {mean}")
```

```
Mean:
X: 4.669245647969052
Y: 4.299806576402321
FFMC: 90.6446808510636
DMC: 110.87234042553195
DC: 547.9400386847191
ISI: 9.021663442940042
temp: 18.88916827852998
RH: 44.28820116054158
wind: 4.017601547388782
rain: 0.02166344294003869
area: 12.847292069632491
```

- ## Finding the Quartile Deviation

```
In [5]: import pandas as pd
        import numpy as np

        # Define a function to calculate quartile deviation
        def quartile_deviation(column):
            Q1 = column.quantile(0.25)
            Q3 = column.quantile(0.75)
            return (Q3 - Q1) / 2

        # Calculate quartile deviation for each numeric column
        quartile_devs = df.select_dtypes(include=np.number).apply(quartile_deviation)

        # Display the results
        print(quartile_devs)
```

```
X         2.000
Y         0.500
FFMC      1.350
DMC      36.900
DC      138.100
ISI       2.150
temp      3.650
RH       10.000
wind      1.100
rain      0.000
area      3.285
dtype: float64
```

- ## Finding the Regression Coefficient

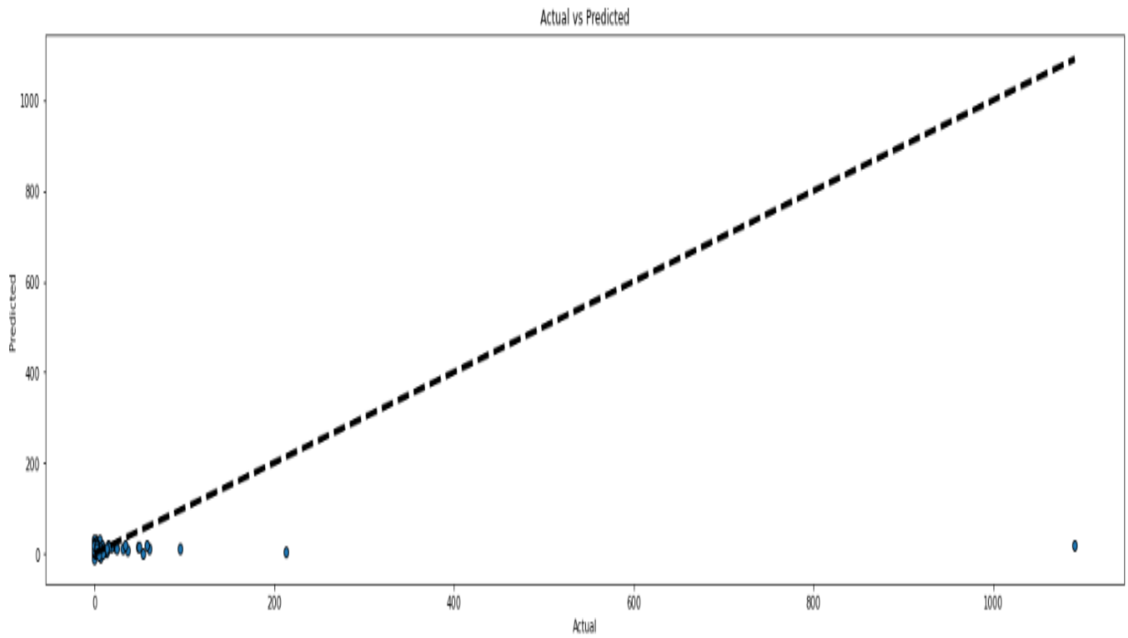
```
In [6]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error, r2_score

        # Prepare the data
        X = df.drop('area', axis=1)
        y = df['area']

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Create and fit the model
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Make predictions and evaluate the model
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        print(f"Mean Squared Error: {mse}")
        print(f"R^2 Score: {r2}")

        # Display the coefficients
        coeff_df = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
        print(coeff_df)
```

```
Mean Squared Error: 11759.94260952891
R^2 Score: 0.0023600333037148147
      Coefficient
X        2.142429
Y        0.060433
FFMC    -0.099682
DMC      0.110716
DC      -0.010381
ISI     -0.297647
temp     0.403704
RH      -0.191226
wind     0.843633
rain    -2.524375
```

- Plotting the Regression Line

```
In [7]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns


X = df.drop('area', axis=1)
y = df['area']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

plt.figure(figsize=(25, 6))
plt.scatter(y_test, y_pred, edgecolors=(0, 0, 0))
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()
```

```
Mean Squared Error: 11759.94260952891
R-squared: 0.0023600333037148147
```



Project I: Salary Prediction using Regression Model

Salary Prediction: Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:
```
sal_data=pd.read_csv('Dataset09-Employee-salary-prediction.csv')
sal_data.head()
```

Out[14]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0

In [15]:
```
sal_data.shape
```

Out[15]:
```
(375, 6)
```

In [16]:
```
sal_data.columns
```

Out[16]:
```
Index(['Age', 'Gender', 'Education Level', 'Job Title', 'Years of Experience',
       'Salary'],
      dtype='object')
```

In [17]:
```
sal_data.dtypes
```

Out[17]:
```
Age                    float64
Gender                  object
Education Level         object
Job Title               object
Years of Experience    float64
Salary                 float64
dtype: object
```

In [18]:
```
sal_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  373 non-null    float64
 1   Gender               373 non-null    object
 2   Education Level      373 non-null    object
 3   Job Title            373 non-null    object
 4   Years of Experience  373 non-null    float64
 5   Salary               373 non-null    float64
dtypes: float64(3), object(3)
memory usage: 17.7+ KB
```

In [19]:
```
sal_data[sal_data.duplicated()]
```

Out[19]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
195	28.0	Male	Bachelor's	Junior Business Analyst	2.0	40000.0
250	30.0	Female	Bachelor's	Junior Marketing Coordinator	2.0	40000.0
251	38.0	Male	Master's	Senior IT Consultant	9.0	110000.0
252	45.0	Female	PhD	Senior Product Designer	15.0	150000.0
253	28.0	Male	Bachelor's	Junior Business Development Associate	2.0	40000.0
254	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
255	44.0	Male	Bachelor's	Senior Software Engineer	14.0	130000.0
256	34.0	Female	Master's	Senior Financial Advisor	6.0	100000.0
257	35.0	Male	Bachelor's	Senior Project Coordinator	9.0	95000.0
258	50.0	Female	PhD	Director of Operations	22.0	180000.0
260	NaN	NaN	NaN	NaN	NaN	NaN
262	46.0	Male	PhD	Senior Data Scientist	18.0	160000.0
281	41.0	Female	Bachelor's	Senior Project Coordinator	11.0	95000.0
287	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
303	45.0	Male	PhD	Senior Data Engineer	16.0	150000.0
306	49.0	Female	Master's	Director of Marketing	21.0	180000.0
307	31.0	Male	Bachelor's	Junior Operations Analyst	3.0	50000.0
309	47.0	Male	Master's	Director of Marketing	19.0	170000.0
310	29.0	Female	Bachelor's	Junior Business Development Associate	1.5	35000.0
311	35.0	Male	Bachelor's	Senior Financial Manager	9.0	100000.0
312	44.0	Female	PhD	Senior Product Designer	15.0	150000.0
313	33.0	Male	Bachelor's	Junior Business Analyst	4.0	60000.0
314	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
315	44.0	Male	Bachelor's	Senior Software Engineer	13.0	130000.0
317	36.0	Male	Bachelor's	Senior Marketing Specialist	8.0	95000.0
328	38.0	Female	Bachelor's	Senior Business Analyst	10.0	110000.0
345	33.0	Male	Bachelor's	Junior Business Analyst	4.0	60000.0
346	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
352	38.0	Female	Bachelor's	Senior Business Analyst	10.0	110000.0
353	48.0	Male	Master's	Director of Marketing	21.0	180000.0
354	31.0	Female	Bachelor's	Junior Business Development Associate	3.0	50000.0
355	40.0	Male	Bachelor's	Senior Financial Analyst	12.0	130000.0
356	45.0	Female	PhD	Senior UX Designer	16.0	160000.0
357	33.0	Male	Bachelor's	Junior Product Manager	4.0	60000.0
358	36.0	Female	Bachelor's	Senior Marketing Manager	8.0	95000.0
359	47.0	Male	Master's	Director of Operations	19.0	170000.0
360	29.0	Female	Bachelor's	Junior Project Manager	2.0	40000.0
361	34.0	Male	Bachelor's	Senior Operations Coordinator	7.0	90000.0
362	44.0	Female	PhD	Senior Business Analyst	15.0	150000.0
363	33.0	Male	Bachelor's	Junior Marketing Specialist	5.0	70000.0
364	35.0	Female	Bachelor's	Senior Financial Manager	8.0	90000.0
365	43.0	Male	Master's	Director of Marketing	18.0	170000.0
366	31.0	Female	Bachelor's	Junior Financial Analyst	3.0	50000.0
367	41.0	Male	Bachelor's	Senior Product Manager	14.0	150000.0
368	44.0	Female	PhD	Senior Data Engineer	16.0	160000.0
369	33.0	Male	Bachelor's	Junior Business Analyst	4.0	60000.0

Dropping Duplicate from the data

In [21]:
```
sal_data1 = sal_data.drop_duplicates(keep = 'first')
```

In [23]:
```
sal_data1.shape
```

Out[23]: (325, 6)

Missing/Null values in each columns:

In [24]:
```
sal_data1.isnull().sum()
```

Out[24]:
```
Age                    1
Gender                 1
Education Level        1
Job Title              1
Years of Experience    1
Salary                 1
dtype: int64
```

Dropping missing values from the data

In [25]:
```
sal_data1.dropna(how='any', inplace= True)
```

```
C:\Users\dekab\AppData\Local\Temp\ipykernel_16688\834867423.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  sal_data1.dropna(how='any', inplace= True)
```

In [26]:
```
sal_data1.shape
```

Out[26]: (324, 6)

Size of the data after dropping duplicate and null values:

In [27]:
```
sal_data1.head()
```

Out[27]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0

In [ ]:

Statistics of numerical columns

In [28]:
```
sal_data.describe()
```

Out[28]:

	Age	Years of Experience	Salary
count	373.000000	373.000000	373.000000
mean	37.431635	10.030831	100577.345845
std	7.069073	6.557007	48240.013482
min	23.000000	0.000000	350.000000
25%	31.000000	4.000000	55000.000000
50%	36.000000	9.000000	95000.000000
75%	44.000000	15.000000	140000.000000
max	53.000000	25.000000	250000.000000

Correlation Matrix among Numerical Column

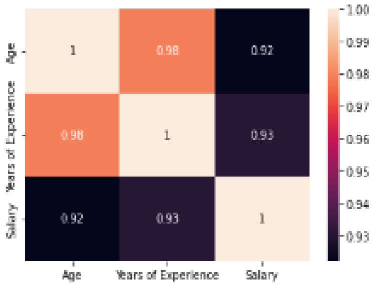
In [29]:
```
corr= sal_data[['Age','Years of Experience','Salary']].corr()
corr
```

Out[29]:

	Age	Years of Experience	Salary
Age	1.000000	0.979128	0.922335
Years of Experience	0.979128	1.000000	0.930338
Salary	0.922335	0.930338	1.000000

In [30]:
```
sns.heatmap(corr,annot= True)
```

Out[30]: <AxesSubplot:>



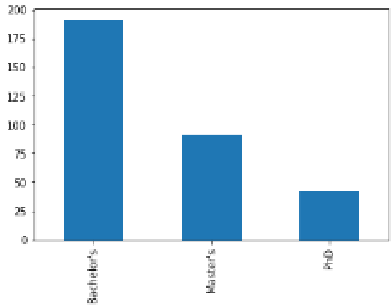
Data Visualisation- Bar chart, Box plot, Histogram

In [31]:
```
1 sal_data1['Education Level'].value_counts()
```

Out[31]:
```
Bachelor's    191
Master's       91
PhD            42
Name: Education Level, dtype: int64
```

In [32]:
```
sal_data1['Education Level'].value_counts().plot(kind= 'bar')
```

Out[32]: <AxesSubplot:>

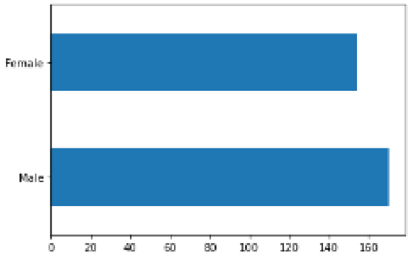


In [33]:
```
sal_data1['Job Title'].value_counts()
```

Out[33]:
```
Director of Operations               9
Director of Marketing                8
Senior Marketing Manager             8
Senior Project Manager               7
Senior Business Analyst              6
                                    ..
Business Development Manager         1
Customer Service Representative      1
IT Manager                           1
Digital Marketing Manager            1
Junior Web Developer                 1
Name: Job Title, Length: 174, dtype: int64
```

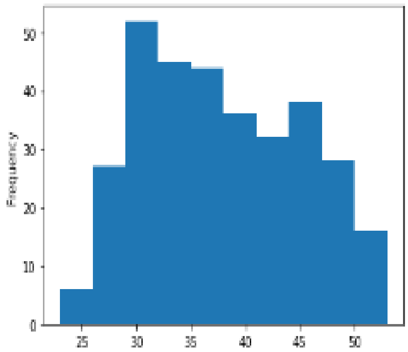
In [35]:
```
sal_data1['Gender'].value_counts().plot(kind= 'barh')
```

Out[35]: <AxesSubplot:>

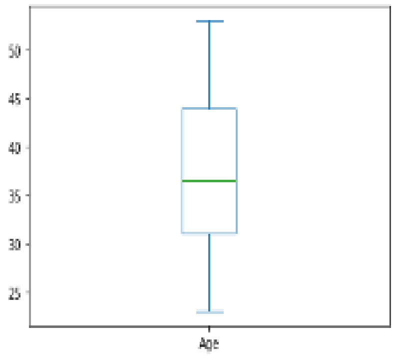


Numerical Variable- Plot Histogram/box plot

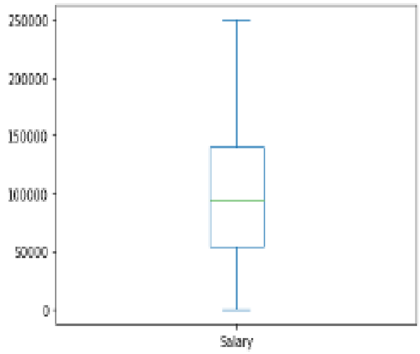
In [36]:
```
sal_data1.Age.plot(kind='hist')
```
Out[36]: <AxesSubplot:ylabel='Frequency'>



In [37]:
```
sal_data1.Age.plot(kind='box')
```
Out[37]: <AxesSubplot:>



In [39]:
```
sal_data1.Salary.plot(kind='box')
```
Out[39]: <AxesSubplot:>



Feature Engineering:

Label encoding

In [42]:
```
from sklearn.preprocessing import LabelEncoder
Label_Encoder=LabelEncoder()
```

In [ ]:
```
sal_data1['Gender_Encode'] = Label_Encoder.fit_transform(sal_data1['Gender'])
```

In [47]:
```
sal_data1['Education Level_Encode'] = Label_Encoder.fit_transform(sal_data1['Education Level'])
```

```
C:\Users\dekab\AppData\Local\Temp\ipykernel_16688\3382122910.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  sal_data1['Education Level_Encode'] = Label_Encoder.fit_transform(sal_data1['Education Level'])
```

In [ ]:
```
sal_data1['Job Title_Encode'] = Label_Encoder.fit_transform(sal_data1['Job Title'])
```

Data after Label Encoding

In [48]:
```
sal_data1.head()
```

Out[48]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Gender_Encode	Job Title_Encode	Education Level_Encode
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0	1	159	0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0	0	17	1
2	45.0	Male	PhD	Senior Manager	15.0	150000.0	1	130	2
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0	0	101	0
4	52.0	Male	Master's	Director	20.0	200000.0	1	22	1

Feature Scaling ¶

In [50]:
```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
```

In [51]:
```
sal_data1['Age_scaled']= std_scaler.fit_transform(sal_data1[['Age']])
sal_data1['Years of Experience_scaled']= std_scaler.fit_transform(sal_data1[['Years of Experience']])
```

```
C:\Users\dekab\AppData\Local\Temp\ipykernel_16688\1712623674.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  sal_data1['Age_scaled']= std_scaler.fit_transform(sal_data1[['Age']])
C:\Users\dekab\AppData\Local\Temp\ipykernel_16688\1712623674.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
```

Data after Scaling

In [52]:
```
sal_data1.head()
```

Out[52]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Gender_Encode	Job Title_Encode	Education Level_Encode	Age_scaled	Years of Experience_scaled
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0	1	159	0	-0.750231	-0.761821
1	28.0	Female	Master's	Data Analyst	3.0	65000.0	0	17	1	-1.307742	-1.063017
2	45.0	Male	PhD	Senior Manager	15.0	150000.0	1	130	2	1.061680	0.744158
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0	0	101	0	-0.192720	-0.460625
4	52.0	Male	Master's	Director	20.0	200000.0	1	22	1	2.037324	1.497148

Dependent and Independent features:

In [53]:
```
x=sal_data1[['Age_scaled','Gender_Encode','Education Level_Encode','Job Title_Encode','Years of Experience_scaled']]
y=sal_data1['Salary']
```

In [54]:
```
x.head()
```

Out[54]:

	Age_scaled	Gender_Encode	Education Level_Encode	Job Title_Encode	Years of Experience_scaled
0	-0.750231	1	0	159	-0.761821
1	-1.307742	0	1	17	-1.063017
2	1.061680	1	2	130	0.744158
3	-0.192720	0	0	101	-0.460625
4	2.037324	1	1	22	1.497148

Splitting the Data into Training and Testing:

In [55]:
```
from sklearn.model_selection import train_test_split
```

In [60]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=42)
```

In [61]:
```
x_train.head()
```

Out[61]:

	Age_scaled	Gender_Encode	Education Level_Encode	Job Title_Encode	Years of Experience_scaled
73	-1.307742	1	0	166	-1.213615
182	0.922302	0	2	155	0.744158
17	0.225413	1	2	116	0.292364
24	0.504169	1	1	37	0.442962
146	0.643547	0	2	115	0.894756

Model Development:

In [62]:
```
from sklearn.linear_model import LinearRegression
```

In [63]:
```
Linear_regression_model = LinearRegression()
```

Model Training:

In [64]:
```
Linear_regression_model.fit(x_train, y_train)
```

Out[64]:
```
LinearRegression()
```

Model Predictions:

In [65]:
```
y_pred_lr= Linear_regression_model.predict(x_test)
y_pred_lr
```

Out[65]:
```
array([117415.91344602, 125562.80742758,  48965.15386167, 128739.34887988,
       106828.49930535,  99654.76748821,  49101.27883652,  57130.71108104,
       166333.69009266,  43112.61060113,  40544.18249367, 122553.217185  ,
       107631.15450848, 155580.48335296,  83652.23602446, 170890.28450907,
        98984.50106226, 109338.33008328,  42267.86835535,  48089.87647812,
        75674.93528581,  64499.29874156,  63619.2494321 ,  31543.41552147,
       188376.92844437,  90340.76921722, 155285.91529198, 160863.57809872,
       185183.73163709,  34741.26224478, 124850.6230462 , 165106.94121635,
        87085.00622186, 155425.69514031, 149190.25441885,  45729.74800187,
        88475.39474629,  92025.62668073,  97997.32557607,  40411.112659  ,
        89995.79796521,  53873.21977084, 108677.48549927,  54590.96778663,
        36497.92729223,  48611.85493217, 129193.72126941,  43102.58902589,
       162383.16672117,  81874.95829259, 157771.0301154 ,  43984.89040816,
        59950.21740617,  94023.81456492,  84929.3880918 ,  60296.00325465,
        91816.87952546,  56177.1258728 ,  75243.32853162, 104701.69952733,
       117279.78847117,  83396.82187583, 177743.76102871,  72275.14427419,
        86307.61361918])
```

In [ ]:
```
df=pd.DataFrame({'y_Actual':y_test,'y_Predicted':y_pred_lr})
df['Error']=df['y_Actual'] - df['y_Predicted']
df['abs_error']=abs(df['Error'])
Mean_absolute_Error=df['abs_error'].mean()
```

Model Evaluation : ¶

In [66]:
```
from sklearn.metrics import accuracy_score,r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Model Accuracy:

In [67]:
```
r2_score(y_test,y_pred_lr)
print(f'Accuracy of the model ={round(r2_score(y_test,y_pred_lr),4)*100}%')
```

```
Accuracy of the model =89.11%
```

Mean Absolute Error:

```
In [68]: round(mean_absolute_error(y_test,y_pred_lr),2)
Out[68]: 10570.79
```

```
In [69]: print(f'Mean Absolute Error ={round(mean_absolute_error(y_test,y_pred_lr),2)}')
Mean Absolute Error =10570.79
```

Mean Squared Error

```
In [71]: mse=round(mean_squared_error(y_test,y_pred_lr),2)
         mse
Out[71]: 205754135.72
```

```
In [72]: print(f'Mean Squared Error={round(mean_squared_error(y_test,y_pred_lr),2)}')
Mean Squared Error=205754135.72
```

Root Mean Squared Error

```
In [73]: print('Root Mean Squared Error(RMSE)=' ,mse ** (0.5))
Root Mean Squared Error(RMSE)= 14344.132449193294
```

Coefficients:

```
In [74]: Linear_regression_model.coef_
Out[74]: array([2.01818940e+04, 7.38907834e+03, 1.54227359e+04, 1.95769562e+01,
                1.92043082e+04])
```

Intercepts:

```
In [76]: Linear_regression_model.intercept_
Out[76]: 86001.49320553194
```

```
In [ ]: Age1= std_scaler.transform([[45]])
        Age =1.061680
        Gender = 1
        Education Level=2
        Job Title = 130
        Years of Experience1 = std_scaler.transform([[15]])
        Years of Experience = 0.744158
        #find the salary=?
```

```
In [ ]: std_scaler.transform([[15]])
```

```
In [ ]: Emp_Salary= Linear_Regression_model.predict([[Age,Gender,Education Level,Job Title,Years of Experience]])
        Emp_Salary
```

```
In [ ]: print("Salary of that Employee with above attributes = ", Emp_Salary[0])
```

By this model, we can calculate the salary of any employee with any attributes(we used the Regression concept to solve the above problem)

Project II: Book Recommender System

jupyter BookRecommenderSystem Last Checkpoint: 10 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
books= pd.read_csv('Books.csv')
users= pd.read_csv('Users.csv')
ratings= pd.read_csv('Ratings.csv')
```

```
C:\Users\dekab\AppData\Local\Temp\ipykernel_6464\3158722078.py:1: DtypeWarning: Columns (3) have mixed types. Specify dtype opt
ion on import or set low_memory=False.
  books= pd.read_csv('Books.csv')
```

In [3]:
```
books.head()
```

Out[3]:

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	Image-URL-M
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.com/images/P/0195153448.0...
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.com/images/P/0002005018.0...
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.com/images/P/0060973129.0...
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.com/images/P/0374157065.0...
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.com/images/P/0393045218.0...

- Checking if there is any Null value

In [6]:
```
print(books.shape)
print(ratings.shape)
print(users.shape)
```

```
(271360, 8)
(1149780, 3)
(278858, 3)
```

In [7]:
```
books.isnull().sum()
```

Out[7]:
```
ISBN                   0
Book-Title             0
Book-Author            1
Year-Of-Publication    0
Publisher              2
Image-URL-S            0
Image-URL-M            0
Image-URL-L            3
dtype: int64
```

In [8]:
```
users.isnull().sum()
```

Out[8]:
```
User-ID          0
Location         0
Age         110762
dtype: int64
```

In [10]:
```
ratings.isnull().sum()
```

Out[10]:
```
User-ID        0
ISBN           0
Book-Rating    0
dtype: int64
```

- Checking if any duplicate attributes are present

In [11]:
```
books.duplicated().sum()
```

Out[11]: 0

In [12]:
```
ratings.duplicated().sum()
```

Out[12]: 0

In [13]:
```
users.duplicated().sum()
```

Out[13]: 0

There are four types of recommender systems namely- Popularity Based,Content Based,Collaborative Filtering based,Hybrid Recommender System. In this project, we will first use Popularity Based Recommender system and then we will use Collaborative Filtering Based Recommender System.

Popularity Based Recommender System

we will display the top 50 books with highest average rating. But we will consider oly those books which have got minimum of 250 votes.

In [19]:
```
ratings_name = ratings.merge(books, on='ISBN')
```

In [20]:
```
ratings_name
```

Out[20]:

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.coı
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.coı
2	6543	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.coı
3	8680	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.coı
4	10314	034545104X	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.coı
...	...	...	...	...	...	...	...	...	
1031131	276688	0517145553	0	Mostly Harmless	Douglas Adams	1995	Random House Value Pub	http://images.amazon.com/images/P/0517145553.0...	http://images.amazon.coı
1031132	276688	1575660792	7	Gray Matter	Shirley Kennett	1996	Kensington Publishing Corporation	http://images.amazon.com/images/P/1575660792.0...	http://images.amazon.coı
1031133	276690	0590907301	0	Triplet Trouble and the Class Trip (Triplet Tr...	Debbie Dadey	1997	Apple	http://images.amazon.com/images/P/0590907301.0...	http://images.amazon.coı
1031134	276704	0679752714	0	A Desert of Pure Feeling (Vintage Contemporaries)	Judith Freeman	1997	Vintage Books USA	http://images.amazon.com/images/P/0679752714.0...	http://images.amazon.coı
				Perplexing					

In [24]:
```
num_rating_df = ratings_name.groupby('Book-Title').count()['Book-Rating'].reset_index()
num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace= True)
num_rating_df
```

Out[24]:

	Book-Title	num_ratings
0	A Light in the Storm: The Civil War Diary of ...	4
1	Always Have Popsicles	1
2	Apple Magic (The Collector's series)	1
3	Ask Lily (Young Women of Faith: Lily Series, ...	1
4	Beyond IBM: Leadership Marketing and Finance ...	1
...	...	...
241066	Ã?Â?lpiraten.	2
241067	Ã?Â?rger mit Produkt X. Roman.	4
241068	Ã?Â?sterlich leben.	1
241069	Ã?Â?stlich der Berge.	3
241070	Ã?Â?thique en toc	2

241071 rows × 2 columns

In [42]:
```
avg_rating_df = ratings_name.groupby('Book-Title').mean()['Book-Rating'].reset_index()
avg_rating_df.rename(columns={'Book-Rating':'avg_ratings'},inplace= True)
avg_rating_df
```

Out[42]:

	Book-Title	avg_ratings
0	A Light in the Storm: The Civil War Diary of ...	2.250000
1	Always Have Popsicles	0.000000
2	Apple Magic (The Collector's series)	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	0.000000

In [43]:
```
popular_df = num_rating_df.merge(avg_rating_df,on= 'Book-Title')
popular_df
```

Out[43]:

	Book-Title	num_ratings	avg_ratings
0	A Light in the Storm: The Civil War Diary of ...	4	2.250000
1	Always Have Popsicles	1	0.000000
2	Apple Magic (The Collector's series)	1	0.000000
3	Ask Lily (Young Women of Faith: Lily Series, ...	1	8.000000
4	Beyond IBM: Leadership Marketing and Finance ...	1	0.000000
...	...	...	...
241066	Ã?Â?lpiraten.	2	0.000000
241067	Ã?Â?rger mit Produkt X. Roman.	4	5.250000
241068	Ã?Â?sterlich leben.	1	7.000000
241069	Ã?Â?stlich der Berge.	3	2.666667
241070	Ã?Â?thique en toc	2	4.000000

241071 rows × 3 columns

In [52]:
```
popular_df=popular_df[popular_df['num_ratings']>=250].sort_values('avg_ratings', ascending=False).head(50)
```

In [53]:
```
popular_df
```

Out[53]:

	Book-Title	num_ratings	avg_ratings
80434	Harry Potter and the Prisoner of Azkaban (Book 3)	428	5.852804
80422	Harry Potter and the Goblet of Fire (Book 4)	387	5.824289
80441	Harry Potter and the Sorcerer's Stone (Book 1)	278	5.737410
80426	Harry Potter and the Order of the Phoenix (Boo...	347	5.501441

In [54]:
```
popular_df.merge(books,on= 'Book-Title').drop_duplicates('Book-Title').shape
```

Out[54]: (50, 10)

In [59]:
```
popular_df= popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Title','Book-Author','Image-URL-M','num_
```

In [60]:
```
popular_df
```

Out[60]:

	Book-Title	Book-Author	Image-URL-M	num_ratings	avg_ratings
0	Harry Potter and the Prisoner of Azkaban (Book 3)	J. K. Rowling	http://images.amazon.com/images/P/0439136350.0...	428	5.852804
3	Harry Potter and the Goblet of Fire (Book 4)	J. K. Rowling	http://images.amazon.com/images/P/0439139597.0...	387	5.824289
5	Harry Potter and the Sorcerer's Stone (Book 1)	J. K. Rowling	http://images.amazon.com/images/P/0590353403.0...	278	5.737410
9	Harry Potter and the Order of the Phoenix (Boo...	J. K. Rowling	http://images.amazon.com/images/P/043935806X.0...	347	5.501441
13	Harry Potter and the Chamber of Secrets (Book 2)	J. K. Rowling	http://images.amazon.com/images/P/0439064872.0...	556	5.183453
16	The Hobbit : The Enchanting Prelude to The Lor...	J.R.R. TOLKIEN	http://images.amazon.com/images/P/0345339681.0...	281	5.007117
17	The Fellowship of the Ring (The Lord of the Ri...	J.R.R. TOLKIEN	http://images.amazon.com/images/P/0345339703.0...	368	4.948370
26	Harry Potter and the Sorcerer's Stone (Harry P...	J. K. Rowling	http://images.amazon.com/images/P/059035342X.0...	575	4.895652
28	The Two Towers (The Lord of the Rings, Part 2)	J.R.R. TOLKIEN	http://images.amazon.com/images/P/0345339711.0...	260	4.880769
39	To Kill a Mockingbird	Harper Lee	http://images.amazon.com/images/P/0446310786.0...	510	4.700000
47	The Da Vinci Code	Dan Brown	http://images.amazon.com/images/P/0385504209.0...	898	4.642539
53	The Five People You Meet in Heaven	Mitch Albom	http://images.amazon.com/images/P/0786868716.0...	430	4.551163
55	The Catcher in the Rye	J.D. Salinger	http://images.amazon.com/images/P/0316769487.0...	449	4.545657
62	The Lovely Bones: A Novel	Alice Sebold	http://images.amazon.com/images/P/0316666343.0...	1295	4.468726
63	1984	George Orwell	http://images.amazon.com/images/P/0451524934.0...	284	4.454225
72	Prodigal Summer: A Novel	Barbara Kingsolver	http://images.amazon.com/images/P/0060959037.0...	253	4.450593
73	Neverwhere	Neil Gaiman	http://images.amazon.com/images/P/0380789019.0...	265	4.449057
78	The Secret Life of Bees	Sue Monk Kidd	http://images.amazon.com/images/P/0142001740.0...	774	4.447028

Collaborative Filtering Based Recommender System

In [61]:
```
ratings_name
```
Out[61]:

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	276725	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.co
1	2313	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.co
2	6543	034545104X	0	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.co
3	8680	034545104X	5	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.co
4	10314	034545104X	9	Flesh Tones: A Novel	M. J. Rose	2002	Ballantine Books	http://images.amazon.com/images/P/034545104X.0...	http://images.amazon.co
...	...	...	...	...	...	...	...	...	
1031131	276688	0517145553	0	Mostly Harmless	Douglas Adams	1995	Random House Value Pub	http://images.amazon.com/images/P/0517145553.0...	http://images.amazon.co
1031132	276688	1575660792	7	Gray Matter	Shirley Kennett	1996	Kensington Publishing Corporation	http://images.amazon.com/images/P/1575660792.0...	http://images.amazon.co
1031133	276690	0590907301	0	Triplet Trouble and the Class Trip (Triplet Tr...	Debbie Dadey	1997	Apple	http://images.amazon.com/images/P/0590907301.0...	http://images.amazon.co
1031134	276704	0679752714	0	A Desert of Pure Feeling (Vintage Contemporaries)	Judith Freeman	1997	Vintage Books USA	http://images.amazon.com/images/P/0679752714.0...	http://images.amazon.co
1031135	276704	0806917695	5	Perplexing Lateral Thinking Puzzles: Scholasti...	Paul Sloane	1997	Sterling Publishing	http://images.amazon.com/images/P/0806917695.0...	http://images.amazon.co

1031136 rows × 10 columns

In [74]:
```
ratings_1 = ratings_name[ratings_name['User-ID'].isin(padhe_likhe_users)]
```
In [78]:
```
y=ratings_1.groupby('Book-Title').count()['Book-Rating']>50
famous_books = y[y].index
```
In [79]:
```
famous_books
```
Out[79]:
```
Index(['1984', '1st to Die: A Novel', '2nd Chance', '4 Blondes',
       'A Bend in the Road', 'A Case of Need',
       'A Child Called \It\": One Child's Courage to Survive"',
       'A Civil Action', 'A Day Late and a Dollar Short', 'A Fine Balance',
       ...
       'Winter Solstice', 'Wish You Well', 'Without Remorse',
       'Wizard and Glass (The Dark Tower, Book 4)', 'Wuthering Heights',
       'Year of Wonders', 'You Belong To Me',
       'Zen and the Art of Motorcycle Maintenance: An Inquiry into Values',
       'Zoya', '\O\" Is for Outlaw"'],
      dtype='object', name='Book-Title', length=679)
```
In [82]:
```
final_ratings= ratings_1[ratings_1['Book-Title'].isin(famous_books)]
```
In [86]:
```
pt=final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rating')
```
In [88]:
```
pt.fillna(0,inplace=True)
```
In [89]:
```
pt
```
Out[89]:

User-ID	254	2276	2766	2977	3363	4017	4385	6251	6323	6543	...	271705	273979	274004	274061	274301	274308	275970	277427	277639	2784
Book-Title																					
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4 Blondes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

- Creating a function *recommend*

In [90]:
```
from sklearn.metrics.pairwise import cosine_similarity
```
In [93]:
```
similarity_scores = cosine_similarity(pt)
```
In [94]:
```
similarity_scores.shape
```
Out[94]:
```
(679, 679)
```
In [108]:
```
def recommend(book_name):
    #index fetch
    index = np.where(pt.index==book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse= True)[1:6]

    for i in similar_items:
        print(pt.index[i[0]])
```
In [109]:
```
pt.index[545]
```
Out[109]:
```
'The Loop'
```

Output:

```
In [108]: def recommend(book_name):
              #index fetch
              index = np.where(pt.index==book_name)[0][0]
              similar_items = sorted(list(enumerate(similarity_scores[index])),key=lambda x:x[1],reverse= True)[1:6]

              for i in similar_items:
                  print(pt.index[i[0]])

In [109]: pt.index[545]

Out[109]: 'The Loop'

In [110]: recommend('The Notebook')

          A Walk to Remember
          The Rescue
          One Door Away from Heaven
          Toxin
          The Five People You Meet in Heaven
```

When we give a book name (in above example,it is-‘The Notebook’),the model will recommend some similar type of book’s name .

Conclusion

In conclusion, our internship experience in Machine Learning has been incredibly rewarding and insightful. Through the projects on salary prediction and book recommendation systems, we've not only applied theoretical knowledge gained from coursework but also developed practical skills in data preprocessing, model development, and evaluation. The Salary Predictor project allowed us to delve into regression techniques and feature engineering, enabling accurate predictions of salary ranges based on various factors. On the other hand, working on the Book Recommender System broadened our understanding of collaborative filtering and its application in personalized recommendations, enhancing user engagement and satisfaction.

Throughout these projects, we encountered challenges that pushed us to think critically and creatively, honing our problem-solving abilities and fostering a deeper appreciation for the iterative nature of machine learning model development. Moreover, collaborating with experienced mentors and peers provided valuable insights and guidance, which significantly contributed to our professional growth.

Looking ahead, we are eager to continue exploring the diverse applications of machine learning in solving real-world problems. The skills and knowledge gained during this internship have laid a solid foundation for our future endeavors in machine learning. We are grateful for the opportunity to have contributed meaningfully to these projects and are excited to apply what we've learned to future challenges in the field.