Exploratory Data Analysis

Dataset Used : Coursera Course Dataset

URL : https://www.kaggle.com/datasets/siddharthm1698/coursera-course-dataset

Data Brief

Course dataset scrapped from Coursera website. This dataset contains mainly 6 columns and 890 course data. The detailed description:

1. **course_title : Contains the course title.**
2. **course_organization : It tells which organization is conducting the courses.**
3. **courseCertificatetype : It has details about what are the different certifications available in courses.**
4. **course_rating : It has the ratings associated with each course.**
5. **course_difficulty : It tells about how difficult or what is the level of the course.**
6. **coursestudentsenrolled : It has the number of students that are enrolled in the course.**

Data Loading and Basic Review

Required Modules

In [58]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps
```

Data Loading and Basic Exploration

In [59]:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [60]:

```
df=pd.read_csv("/kaggle/input/coursera-course-dataset/coursea_data.csv")
df.head()
```

In [61]:

```
df=df.drop("Unnamed: 0",axis=1)
df.info()
```

So, 1 numarical object only. But, we can turn some others to numarical too.

In [62]:

```
df.describe()
```

Mean course rating is 4.677329. Quite high, as the rating can be given from 0-5. Minimum is 3.3, highest is 5 - proves so.

Initial plan for data exploration

Data Exploration

1. **Ploting course_rating to get a overview of the distribution.**
2. **analyzing course Certificate types values.**

Data Cleaning

1. **Deleting first Unnamed column**
2. **Deleting course name - not necessary now; as all the values are unique**

Data Exploration

Basic Rating distribution :

In [63]:

```
# Ploting course_rating to get a overview of the distribution.
plt.boxplot(df['course_rating'])
```

In [64]:

```
# Ploting course_rating to get a overview of the distribution.
df['course_rating'].hist()
```

Findings:

Average course rating is quite higher, compared to lowest and maximum value.

Rating distribution per course difficulty :

In [65]:

```
g = df.groupby('course_difficulty')['course_rating']
fig, axes = plt.subplots(g.ngroups, sharex=True, figsize=(4, 6))

for i, (type,rating) in enumerate(g):
    ax = rating.plot.hist('course_rating',ax=axes[i], legend=False,title=type)
fig.tight_layout()
```

Insight:

Advanced courses' rating has some ups-and downs; maybe due to low frequency.
Beginner course has distribution quite similiar to total rating chart.
Intermidiate course's rating top is not as sharp of others, that may say - as the participants has some knowledge on the topic, they can judge better and being critical.

Rating distribution per course type :

In [66]:

```
g = df.groupby('course_Certificate_type')['course_rating']
fig, axes = plt.subplots(g.ngroups, sharex=True, figsize=(4, 6))

for i, (type,rating) in enumerate(g):
    ax = rating.plot.hist('course_rating',ax=axes[i], legend=False,title=type,bins=10)
fig.tight_layout()
```

In [67]:

```
g.describe()
```

Findings and Insight:

1. **Specializations has lower mean value than courses, but the distribution is interesting. specialization has good distribution values on right, but normal courses are on left.**

Combined

In [68]:

```
g = df.groupby(['course_difficulty','course_Certificate_type'])['course_rating']
fig, axes = plt.subplots(g.ngroups, sharex=True, figsize=(4, 20))

for i, (type,rating) in enumerate(g):
    axes[i].set_ylim(0, 100)
    ax = rating.plot.hist('course_rating',ax=axes[i], legend=False,title=type[0]+"-"+typ
e[1].lower(),bins=10)
fig.tight_layout()
```

Analyzing course Certificate types values.

In [69]:

```
df.groupby('course_difficulty').course_difficulty.value_counts().unstack().plot.barh()
```

In [70]:

```
df.groupby('course_Certificate_type').course_Certificate_type.value_counts().unstack().p
lot.barh()
```

Data Cleaning

1. **Deleting first Unnamed column**
2. **Deleting course name - not necessary now; as all the values are unique**

In [71]:

```
df=df.drop(['course_title'],axis=1)
```

Feature Engineering

1. **Modifying course_students_enrolled column**

In [72]:

```
df_fe1=df.copy()
```

In [73]:

```
def course_students_enrolled_modifier(x):
    return x[:-2]
```

In [74]:

```
df_fe1['course_students_enrolled_modified']=df_fe1['course_students_enrolled'].apply(cour
se_students_enrolled_modifier)
df_fe1['course_students_enrolled_modified']=df_fe1['course_students_enrolled_modified'].a
pply(pd.to_numeric)
df_fe1 =df_fe1.drop(['course_students_enrolled'],axis=1)
df_fe1
```

1. **Modifying course_difficulty column to numarical**

In [75]:

```
def course_difficulty_modifier(x):
    if x=="Beginner":
        return "0"
    elif x=="Intermediate":
        return "1"
    elif x=="Mixed":
        return "0.5"
    elif x=="Advanced":
        return "2"
    else:
        return "0"
"""as most courses are beginner level, we are assuming undefined will be beginner too."""
```

In [76]:

```
df_fe1['course_difficulty_modified']=df_fe1['course_difficulty'].apply(course_difficulty_
modifier)
df_fe1['course_difficulty_modified']=df_fe1['course_difficulty_modified'].apply(pd.to_num
eric)
df_fe1 =df_fe1.drop(['course_difficulty'],axis=1)
df_fe1
```

Data Exploration of newly engineered columns

In [77]:

```
df_fe1[['course_difficulty_modified','course_students_enrolled_modified']].describe()
```

course_students_enrolled_modified has some empty columns, so we have to fill them.

In [78]:

```
df_fe1[['course_students_enrolled_modified']].plot.hist()
```

so , most of the frequencies are in between 0-10, so, using average-1; so avoid the effect of outliers.

In [79]:

```
df_fe1['course_students_enrolled_modified'].fillna((df_fe1['course_students_enrolled_modi
fied'].mean()-1), inplace=True)
df_fe1[['course_difficulty_modified','course_students_enrolled_modified']].describe()
```

In [80]:

```
df_numaric=df_fe1.select_dtypes(include=np.number)
```

Finding relation between columns

In [81]:

```
corrM = df_numaric.corr()
corrM
```

In [82]:

```
df_numaric.plot.scatter(x='course_rating', y='course_difficulty_modified',c='DarkBlue')
```

Findings :

No effective coorelation.

Key Findings and Insights

1. Average course rating is quite higher, compared to lowest and maximum value. So, the cours quality is being maintained.
2. Advanced courses' rating has some ups-and downs; maybe due to low frequency.
3. Beginner course has distribution quite similiar to total rating chart, as big portion of the data is from them, and he number of beginner level courses are high.
4. Intermidiate course's rating top is not as sharp of others, that may say - as the participants has some knowledge on the topic, they can judge better and being critical.
5. Specializations has lower mean value than courses, but the distribution is interesting. specialization has good distribution values on right, but normal courses are on left.
6. No effective coorelation between course_difficulty,course_students_enrolled, course rating.

In [189]:

```
# Hypothesis Testing
```

Dataset Used : Coursera Course Dataset

URL : https://www.kaggle.com/datasets/siddharthm1698/coursera-course-dataset

Featured Engineered Data : https://www.kaggle.com/code/azminetoushikwasi/coursera-eda-prep-viz-fe-with-analytics-insights/notebook

Data Brief

Course dataset scrapped from Coursera website. This dataset contains mainly 6 columns and 890 course data. The detailed description:

1. **course_title : Contains the course title.**
2. **course_organization : It tells which organization is conducting the courses.**
3. **courseCertificatetype : It has details about what are the different certifications available in courses.**
4. **course_rating : It has the ratings associated with each course.**
5. **course_difficulty : It tells about how difficult or what is the level of the course.**
6. **coursestudentsenrolled : It has the number of students that are enrolled in the course.**

In [190]:

```
## Data import and Coorelation Matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sps

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

df=pd.read_csv("coursera_data_FEd.csv")
df=df.drop("Unnamed: 0",axis=1)
```

Sampling example

In [191]:

```
sample=df.sample(40,random_state=1)
sample.describe()
```

Out[191]:

	course_rating	course_students_enrolled_modified	course_difficulty_modified
count	40.000000	40.000000	40.000000
mean	4.675000	7.162798	0.325000
std	0.151488	8.651666	0.460629
min	4.200000	1.000000	0.000000
25%	4.600000	2.000000	0.000000
50%	4.700000	5.000000	0.000000
75%	4.800000	8.000000	0.500000

	course_rating	course_students_enrolled_modified	course_difficulty_modified
max	4.900000	38.000000	2.000000

Hypothesis Formulation

Hypothesis 01:

Null hypothesis: Equal or less than 50% enrolled courses are beginner level courses.

- **Test Method: z statstic**
- **Significance Level: 8%**

Hypothesis 02:

Null hypothesis: Coursera has a average course rating of more than 4.5.

Hypothesis 03:

Null hypothesis: University courses has more average rating by 0.2 from non-university courses.

Conducting a formal significance test for one of the hypotheses and discuss the results

Testing for Hypothesis 01:

Necessary Data

- **H_0: $\pi \leq 0.50$**
- **H_1: $\pi > 0.50$**
- **$\alpha = 0.08$**
- **Test Method: z statstic; $z = (p-\pi)/\sigma_p$, where $\sigma_p = \text{sqrt}(\pi(1-\pi)/n)$**

In [192]:

```
pi=0.5
sigma=0.08
```

Calculating P

In [193]:

```
sample_size=len(df)
sample_size
```

Out[193]:

```
891
```

so, total sample size = 891

In [194]:

```
# P, the value of sample statistic
positives= df[df['course_difficulty_modified']==0]['course_rating'].count()
positives
```

Out[194]:

```
487
```

number of courses with rating more than 4.5 = 745

In [195]:

```
P=positives/sample_size
P
```

Out[195]:

```
0.5465768799102132
```

Now, we will determine The value of σ_p, where σ_p=sqrt(π(1-π)/n)

In [196]:

```
import math

#defining meu_p function
def meu_p (pi,sample_size):
    temp=pi*(1-pi)/sample_size
    return math.sqrt(temp)
meu_p (pi,sample_size)
```

Out[196]:

```
0.016750630254320203
```

In [197]:

```
#defining z_statistic function

def z_stat(pi,p,sample_size):
    return (p-pi)/meu_p(pi,sample_size)
```

In [198]:

```
## Applying
z_stat(pi,P,sample_size)
```

Out[198]:

```
2.7806046222171505
```

In [201]:

```
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "http://www.z-table.com/uploads/2/1/7/9/21795380/8573955.png?759")
```

Out[201]:

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319

1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

Probability is approximately 0.998; But we wanted to calculate the probability to the right of z (because we are interested in obtaining the probability value that falls in the rejection region or critical region), i.e.

In [202]:

```
1-0.998
```

Out[202]:

```
0.0020000000000000018
```

Aplha is 0.05 So, the null hypothesis is rejected.

More than 50% students get enrolled in Beginner level courses.

Suggestions for next steps in analyzing this data

- **Testing other hypotheses.**
- **Analyze university based data.**
- **Try to group the courses to related subjects, based on subject name - keywords and see if any subject/field is performing better than others.**

The quality of this data set and a request for additional data if needed

- **Data quality is good, but data is not well distributed in various categories.**
- **The coirse-rating section is highly one-sided.**
- **Student enrollment number could be given in number, instead of string.**
- **Course length and these type info would have helped more.**

Data Request:

- **Require more data on some categories (advanced and so) to analyse far more better.**
- **More data means more accurate result. For a large platfrom like Cousera, we need more data and meta-data; like date-time of course launch, date of records and so on.**