Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2021-2022)
Data Analytics

Experiment: Exploratory Data Analysis (EDA)

Name: Sneha Ghuge **UID:** 2019110015 **BE ETRX** **DA LAB 3**

Aim: Analyze statistical data.

Objective:

Perform statistical data analysis such as: Estimators of the main statistical measures (mean, variance, standard deviation, covariance correlation, standard error), Main distributions (Normal distribution, chi-square distribution), Hypothesis testing, pairwise association (Pearson correlation test, t-test, ANOVA), Non-parametric test (Spearman rank)

CODE & OUTPUT:

```
import numpy as np
import pandas as pd
from scipy.stats import norm, chi2, pearsonr, stats, spearmanr, f_oneway
```

```
data = pd.read_csv('Social_Network_Ads.csv')
```

ESTIMATING THE MODEL PARAMETERS LIKE MEAN, MEDIAN, MODE, SD, CORRELATION ETC

```
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
data.describe().T
#calculating the five point summary of the data and .T is used to invert the table
```

	count	mean	std	min	25%	50%	75%	max
User ID	400.0	1.569154e+07	71658.321581	15566689.0	15626763.75	15694341.5	15750363.0	15815236.0
Age	400.0	3.765500e+01	10.482877	18.0	29.75	37.0	46.0	60.0
EstimatedSalary	400.0	6.974250e+04	34096.960282	15000.0	43000.00	70000.0	88000.0	150000.0
Purchased	400.0	3.575000e-01	0.479864	0.0	0.00	0.0	1.0	1.0

```
# printing more in detail information about the data that includes the
# datatype of the columns, null values etc
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
data.shape
```

```
(400, 5)
```

```
# checking null values
data.isna().any()
```

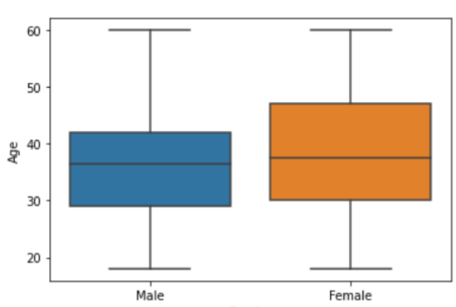
```
User ID            False
Gender             False
Age                False
EstimatedSalary    False
Purchased          False
dtype: bool
```

```
# no of unique datapoints in every column
for col in data:
  space =  ' '*(30-len(col))
  print(col,space,len(data[col].unique()) )
```

```
User ID                          400
Gender                           2
Age                              43
EstimatedSalary                  117
Purchased                        2
```

```
import seaborn as sns
# plotting a boxplot distribution considering the age and gender
sns.boxplot(x="Gender", y="Age", data=data)
```

```
<AxesSubplot:xlabel='Gender', ylabel='Age'>
```

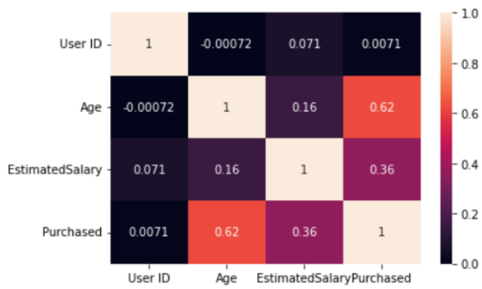


```python
corr = data.corr()
corr #printing the correlation score in a tabular format
```

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

```python
# plotting and displaying correlation heatmap
sns.heatmap(corr, annot=True)
```

```
<AxesSubplot:>
```



1) Unbiased standard error of the mean

```python
# The standard error of the mean is the standard deviation of the sampling distribution
# of the mean. In other words it is the standard deviation of a large number of sample
# means of the same sample size drawn from the same population.
# The term standard error of the mean is commonly (though imprecisely) shortened to just standard error.
# Thus the terms 'standard error of the mean', 'standard deviation of the mean' and 'standard error' may all mean e
data.sem() # find standard error of the mean of all the columns
```

```
C:\Users\91744\AppData\Local\Temp\ipykernel_21352\4087429978.py:6: FutureWarning: Dropping of nuisance columns in D
uctions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid
ore calling the reduction.
  data.sem() # find standard error of the mean of all the columns
```

```
User ID            3582.916079
Age                   0.524144
EstimatedSalary    1704.848014
Purchased             0.023993
dtype: float64
```

2) Mode value for all the axis

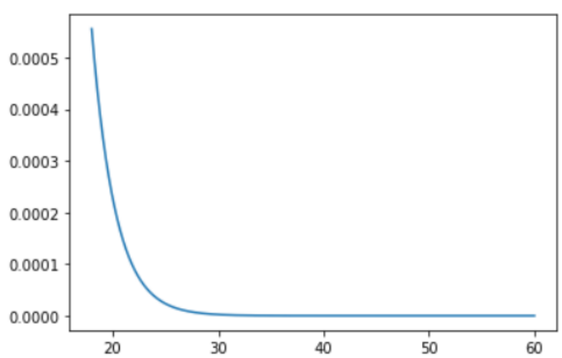
```
for i in data:
    print('\n\n',i,'\n', data[i].mode())
```

```
 User ID
 0      15566689
1      15569641
2      15570769
3      15570932
4      15571059
         ...
395    15813113
396    15814004
397    15814553
398    15814816
399    15815236
Name: User ID, Length: 400, dtype: int64


 Gender
 0    Female
Name: Gender, dtype: object


 Age
 0    35
Name: Age, dtype: int64


 EstimatedSalary
 0    72000
Name: EstimatedSalary, dtype: int64
```

3) Chi-Square Distribution of any axis

```
range = np.arange(data['Age'].min(), data['Age'].max(), 0.001)
plt.plot(range, chi2.pdf(range, df=4))
```

```
[<matplotlib.lines.Line2D at 0x2176ea06a30>]
```



4) Calculating the regression coefficients

```
# Simple Linear Regression
#Load function from sklearn
from sklearn import linear_model

# Create linear regression object
regr = linear_model.LinearRegression()

y = data['Age']
x = data[['EstimatedSalary','Purchased']]

# Train the model using the training sets
regr.fit(x,y)
```

```
LinearRegression()
```

```
regr.coef_
```

```
array([-2.48185110e-05,  1.42363642e+01])
```

```
regr.intercept_
```

```
34.29640478947125
```

Contingency Table

In order to compute the Chi-square test statistic, we would need to construct a contingency table.

We can do that using the ‘crosstab’ function from pandas:

```
data['Purchased'].value_counts()
```

```
0    257
1    143
Name: Purchased, dtype: int64
```

```
data['EstimatedSalary'].value_counts()
```

```
72000     12
80000     11
79000     10
75000      9
71000      9
          ..
123000     1
37000      1
115000     1
148000     1
139000     1
Name: EstimatedSalary, Length: 117, dtype: int64
```

```
ct=pd.crosstab(data.Purchased, data.EstimatedSalary, margins=True)
```

```python
ct=pd.crosstab(data.Purchased, data.EstimatedSalary, margins=True)
```

```python
ct
```

EstimatedSalary	15000	16000	17000	18000	19000	20000	21000	22000	23000	25000	...	141000	142000	143000	144000	146000	147000	148000
Purchased																		
0	4	2	3	4	2	3	1	3	3	3	...	1	0	0	0	0	0	0
1	0	0	0	0	0	2	1	2	4	1	...	1	1	2	4	2	1	1
All	4	2	3	4	2	5	2	5	7	4	...	2	1	2	4	2	1	1

3 rows × 118 columns

```python
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Chi-squared test: First, compute both the **observation frequency** and expected frequency.

```python
obs = np.append(ct.iloc[0][0:6].values, ct.iloc[1][0:6].values)
obs
```

```
array([4, 2, 3, 4, 2, 3, 0, 0, 0, 0, 0, 2], dtype=int64)
```

```python
row_sum = ct.iloc[0:2,6].values
exp = []

for ko in row_sum:
    for val in ct.iloc[2,0:6].values:
        exp.append(val * ko / ct.loc['All', 'All'])
exp
```

```
[0.01,
 0.005,
 0.0075,
 0.01,
 0.005,
 0.0125,
 0.01,
 0.005,
 0.0075,
 0.01,
 0.005,
 0.0125]
```

```python
chi_sq_stats = ((obs - exp)**2/exp).sum()
chi_sq_stats
```

```
7000.100000000001
```

```python
dof = (len(row_sum)-1)*(len(ct.iloc[2,0:6].values)-1)
dof
```

```
5
```

```python
1 - stats.chi2.cdf(chi_sq_stats, dof)
```

```
0.0
```

PEARSON CORRELATION TEST

Only applicable for numerical data
"r" is the correlation coefficient

1) r takes value between -1 (negative correlation) and 1 (positive correlation).
2) r = 0 means no correlation.
3) Can not be applied to ordinal variables.
4) The sample size should be moderate (20-30) for good estimation.
5) Outliers can lead to misleading values means not robust with outliers.

```
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
list1 = data['Age']
list2 = data['Purchased']
# Apply the pearsonr()
corr, _ = pearsonr(list1, list2)
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.622
```

Here after calculating the correlation between Age and Purchased we can see that there is a good amount of positive dependence between them since the coefficient is a somewhat high positive value

```
list1 = data['EstimatedSalary']
list2 = data['Purchased']
# Apply the pearsonr()
corr, _ = pearsonr(list1, list2)
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.362
```

Here after calculating the correlation between EstimatedSalary and Age we can see that there is a good amount of positive dependence between them since the coefficient is a somewhat high positive value and it also explains the fact that experience is equivalent to increase in pay

```
list1 = data['EstimatedSalary']
list2 = data['Age']
# Apply the pearsonr()
corr, _ = pearsonr(list1, list2)
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.155
```

TWO SAMPLE T TEST

```
data_group1 = data["Age"][0:20]
data_group2 = data["Age"][20:40]
# Print the variance of both data groups
print(np.var(data_group1), np.var(data_group2))
print(np.var(data_group1)/np.var(data_group2))
```

```
86.7475 85.74749999999999
1.0116621475844778
```

```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

```
Ttest_indResult(statistic=-1.8253713696450706, pvalue=0.07581240895163492)
```

SPEARMANS CORRELATION TEST

In Spearman rank correlation instead of working with the data values themselves (as discussed in Correlation coefficient), it works with the ranks of these values. The observations are first ranked and then these ranks are used in correlation.
The Spearman rank-order correlation is a statistical procedure that is designed to measure the relationship between two variables on an ordinal scale of measurement.

The hypothesis that we build in case of Spearmans test is : (Assumption alpha=0.05)

H0 => p > alpha (Samples are uncorrelated)

HA => p<= alpha (Samples are correlated)

```
data1=data['EstimatedSalary']
data2=data['Age']
coef, p = spearmanr(data1, data2)
print('Spearmans correlation coefficient: %.3f' % coef)
```

```
Spearmans correlation coefficient: 0.125
```

If we compare this correlation with the above calculated in pearson test we can see that both point towards the fact that Age and EstimatedSalary are positively correlated

```
data1=data['Age']
data2=data['Purchased']
coef, p = spearmanr(data1, data2)
print('Spearmans correlation coefficient: %.3f' % coef)
```

```
Spearmans correlation coefficient: 0.612
```

ONE WAY ANOVA TEST

One-Way ANOVA in Python: One-way ANOVA (also known as "analysis of variance") is a test that is used to find out whether there exists a statistically significant difference between the mean values of more than one group.

The hypothesis that we build in case of Spearmans test is : (Assumption alpha=0.05)

=> H0 (null hypothesis): μ1 = μ2 = μ3 = … = μk (It implies that the means of all the population are equal)
=> H1 (null hypothesis): It states that there will be at least one population mean that differs from the rest

Though performing the ANOVA test does not really make any sense in this context but just for performing the working I will consider the Age attribute and distribute it into groups of 4 with 5 entries in each

```
performance1 = data["Age"][0:5]
performance2 = data["Age"][5:10]
performance3 = data["Age"][10:15]
performance4 = data["Age"][15:20]

# Conduct the one-way ANOVA
f_oneway(performance1, performance2, performance3, performance4)
```

```
F_onewayResult(statistic=9.619909502262445, pvalue=0.0007222893679483783)
```

Conclusion:

Successfully performed statistical data analysis such as: Estimators of the main statistical measures (mean, variance, standard deviation, covariance correlation, standard error), Main distributions (Normal distribution, chi-square distribution), Hypothesis testing, pairwise association (Pearson correlation test, t-test, ANOVA), Non-parametric test (Spearman rank)