19SE02IT058

AIM:- Chi square

- **Chi-Squared Goodness-Of-Fit Test**

```
print("Output")

import numpy as np import
pandas as pd
import·scipy.stats·as·stats
national·=·pd.DataFrame(["white"]*100000·+·["hispanic"]*60000·+\
························["black"]*50000·+·["asian"]*15000·+·["other"]*35000)
···········

minnesota = pd.DataFrame(["white"]*600 + ["hispanic"]*300 + \
["black"]*250 +["asian"]*75 + ["other"]*150)

national_table = pd.crosstab(index=national[0], columns="count")
minnesota_table = pd.crosstab(index=minnesota[0], columns="count")

print( "National")
print(national_table)
print(" ") print(
"Minnesota")
print(minnesota_table)
```

```
Output  National
col_0
count          0
asian
15000      black
50000   hispanic
60000      other
35000      white
100000
  Minnesota
col_0
count 0
asian
75 black
250 hispanic
300 other
150 white
600
```

```
print("Output") observed = minnesota_table national_ratios =

national_table/len(national)  # Get population ratios expected =

national_ratios * len(minnesota)   # Get expected counts
```

```
chi_squared_stat = (((observed-expected)**2)/expected).sum()

print(chi_squared_stat)

     Output col_0 count
     18.194805 dtype:
     float64


print("Output")

crit = stats.chi2.ppf(q = 0.95, # Find the critical value for 95% confidence*
df = 4)   # Df = number of variable categories - 1

print("Critical value") print(crit)

p_value = 1 - stats.chi2.cdf(x=chi_squared_stat,  # Find the p-value
df=4) print("P value") print(p_value)

     Output
     Critical value
     9.487729036781154
     P value
     [0.00113047]


print("Output")

stats.chisquare(f_obs= observed,   # Array of observed counts
f_exp= expected)   # Array of expected counts

     Output
     Power_divergenceResult(statistic=array([18.19480519]), pvalue=array([0.00113047]))
```

- **Chi-Squared Test of Independence**

```
print("Output") np.random.seed(10)

# Sample data randomly at fixed probabilities voter_race = np.random.choice(a=
["asian","black","hispanic","other","white"],                              p
= [0.05, 0.15 ,0.25, 0.05, 0.5],                               size=1000)

# Sample data randomly at fixed probabilities
voter_party = np.random.choice(a= ["democrat","independent","republican"],
p = [0.4, 0.2, 0.4],

                              size=1000)

voters = pd.DataFrame({"race":voter_race,

"party":voter_party}) voter_tab = pd.crosstab(voters.race, voters.party,

margins = True) voter_tab.columns =
["democrat","independent","republican","row_totals"] voter_tab.index =

["asian","black","hispanic","other","white","col_totals"]

observed = voter_tab.iloc[0:5,0:3]   # Get table without totals for later use
voter_tab
```

Output

	democrat	independent	republican	row_totals
asian	21	7	32	60
black	65	25	64	154
hispanic	107	50	94	251
other	15	8	15	38
white	189	96	212	497
col_totals	397	186	417	1000

```
print("Output")
expected     =          np.outer(voter_tab["row_totals"][0:5],

voter_tab.loc["col_totals"][0:3])    /    1000    expected    =

pd.DataFrame(expected)

expected.columns = ["democrat","independent","republican"]

expected.index = ["asian","black","hispanic","other","white"] expected
```

Output

	democrat	independent	republican
asian	23.820	11.160	25.020
black	61.138	28.644	64.218
hispanic	99.647	46.686	104.667
other	15.086	7.068	15.846
white	197.309	92.442	207.249

```
print("Output") chi_squared_stat = (((observed-

expected)**2)/expected).sum().sum() print(chi_squared_stat)
```

```
Output
7.169321280162059
```

```
print("Output")

crit = stats.chi2.ppf(q = 0.95, # Find the critical value for 95% confidence*
print("Critical value")
print(crit)

p_value = 1 - stats.chi2.cdf(x=chi_squared_stat,  # Find the p-value
df=8) print("P value") print(p_value)
```

```
     Output
     Critical value
     15.50731305586545
     P value
     0.518479392948842
```