statistics-lab-10-assignment

November 30, 2023

Q1.Maximize Z = 3x + 4y, subject to the constraints are x + y   4, x   0 and y   0. Solve using numpy.

[ ]:
```
import numpy as np
from scipy.optimize import linprog

#Z = 3x + 4y
c_max = np.array([-3,-4])

#x+y=<=4
#b
A = np.array([[1,1]])
B = np.array([4])
bounds = [(0,None),(0,None)]

result_max = linprog(c_max, A_ub=A, b_ub=B, bounds=bounds)
print("Max optimal value", result_max.x)
print("Max optimal objective value",-result_max.fun)
```

```
Max optimal value [0. 4.]
Max optimal objective value 16.0
```

Q2. Maximize Z=5x+3y, subject to constraints 3x+5y 15, 5x+2y 10, x 0 and y 0. Solve using numpy.

[ ]:
```
import numpy as np
from scipy.optimize import linprog

c_min = np.array([5,3])
A = np.array([[3,5],[5,2]])
B = np.array([15,10])
bounds = [(0,None),(0,None)]

c_max = -c_min
result_min = linprog(c_max, A_ub=A, b_ub=B, bounds=bounds)
print("Max optimal value", result_min.x)
print("Max optimal objective value",-result_min.fun)
```

```
Max optimal value [1.05263158 2.36842105]
```

```
Max optimal objective value 12.36842105263158
```

Q3.What do you meant by factor analysis and its types.

Ans=Factor Analysis (FA) is a statistical method used to understand relationships among multiple variables by identifying underlying, unobservable factors that explain patterns of correlations between these variables. It aims to simplify complex data by reducing the number of variables into a smaller set of latent factors.

Types of Factor Analysis:

a) Exploratory Factor Analysis (EFA): Explores data to identify underlying patterns without preconceived notions about relationships between variables and factors. It helps discover the number of factors and their associations with observed variables.

b) Confirmatory Factor Analysis (CFA): Tests predefined hypotheses or models about relationships between variables and factors. It verifies if observed data aligns with the proposed model.

C) Principal Component Analysis (PCA): While not technically FA, PCA extracts principal components (linear combinations of variables) to capture most variance without considering underlying factors.

Q4.Do factor Analysis on iris dataset.

[ ]:
```python
import numpy as np
import pandas as pd
from sklearn.decomposition import FactorAnalysis
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
feature_names = iris.feature_names

factor_analysis = FactorAnalysis(n_components=2, random_state=42)
X_fa = factor_analysis.fit_transform(X)

factor_loadings = pd.DataFrame(factor_analysis.components_.T,␣
 ↪columns=['Factor1', 'Factor 2'], index=feature_names)

total_variance = np.var(X, axis=0).sum()
explained_variance = np.var(X_fa, axis=0).sum()

explained_variance_ratio = explained_variance/total_variance

df_fa = pd.DataFrame(data = X_fa, columns = ['Factor 1', 'Factor 2'])
df_fa['Target'] = iris.target_names[iris.target]
df_fa
```

[ ]:

	Factor 1	Factor 2	Target
0	-1.327617	-0.561311	setosa
1	-1.337639	-0.002798	setosa

```
2   -1.402815  0.306349     setosa
3   -1.301043  0.718827     setosa
4   -1.333424 -0.364589     setosa
..        …         …          …
145  0.844681 -0.540558  virginica
146  0.707514  0.247219  virginica
147  0.826875 -0.132014  virginica
148  0.922997  0.635225  virginica
149  0.735404  1.084134  virginica

[150 rows x 3 columns]
```

[ ]:
```
explained_variance_ratio
```

[ ]: 0.41885496288023333

[ ]:
```
!pip install factor_analyzer
```

Q5.Do factor analyses of the bfi.csv file and find the eigen values. Based on those select important factors.

[4]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('bfi_1.csv')

X = df.drop(columns=['Unnamed: 0', 'gender', 'education', 'age'],␣
 ↪axis=1,inplace=True)
df.dropna(inplace = True)

X_std = StandardScaler().fit_transform(df.values)
fa = FactorAnalyzer(rotation=None)
fa.fit(X_std)

#Eigenvalues
ev, v = fa.get_eigenvalues()
print("EigenValues: ",ev)
```

```
EigenValues:  [5.13431118 2.75188667 2.14270195 1.85232761 1.54816285 1.07358247
 0.83953893 0.79920618 0.71898919 0.68808879 0.67637336 0.65179984
 0.62325295 0.59656284 0.56309083 0.54330533 0.51451752 0.49450315
 0.48263952 0.448921   0.42336611 0.40067145 0.38780448 0.38185679
 0.26253902]
```

Q6.A company manufactures two products X and Y, which require, the following

resources. The resources are the capacities machine M1, M2, and M3. The available capacities are 50, 25, and 15 hours respectively in the planning period. Product X requires 1 hour of machine M2 and 1 hour of machine M3. Product Y requires 2 hours of machine M1, 2 hours of machine M2 and 1 hour of machine M3. The profit contribution of products X and Y are Rs.5/- and Rs.4/- respectively. Solve using any method.

Ans= X requires 0 hours of M1, 1 hour of M2, and 1 hour of M3.

YY requires 2 hours of M1, 2 hours of M2, and 1 hour of M3.

Available capacities: 2X+2Y 502X+2Y 50 (for M2), X+Y 25X+Y 25 (for M3), and Y 15Y 15 (for M3).

Non-negativity constraints: X,Y 0X,Y 0

[ ]:
```python
import numpy as np
from scipy.optimize import linprog

# Z = 5X + 4Y
c = [-5, -4]

# Coefficients for M2 in X and Y (2X + 2Y <= 50),for M3 in X and Y (X + Y <=␣
↪25),Coefficients for M3 in Y (Y <= 15)
A = [[0, 2],[1, 1],[0, 1] ]

# Available capacities for M2, M3, M3
b = [50, 25, 15]

result = linprog(c, A_ub=A, b_ub=b)

if result.success:
    print("Optimal quantities:")
    print(f"Product X: {result.x[0]} units")
    print(f"Product Y: {result.x[1]} units")
    print(f"Maximum Profit: {-result.fun} Rs.")
else:
    print("Optimization failed. Check constraints.")
```

```
Optimal quantities:
Product X: 25.0 units
Product Y: 0.0 units
Maximum Profit: 125.0 Rs.
```

Q7. Consider the story of wolf and farmer. State the facts and proof from it. Construct confusion matrix.

Ans=

a) True Positives (TP): Instances where the boy cried "Wolf," and there was indeed a wolf threatening the sheep. (None in the story) *b) True Negatives (TN): Instances where the boy didn't cry "Wolf," and there was no actual wolf. (Initial situation before the real wolf appeared)*

c) False Positives (FP): Instances where the boy falsely cried "Wolf," and there was no actual wolf. (The repeated false alarms by the boy)

d) False Negatives (FN): Instances where the boy didn't cry "Wolf," but there was a real wolf threatening the sheep. (The final situation when the real wolf appeared)