

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра «Системи штучного інтелекту»



Лабораторна робота №2
з курсу «Аналітичні сховища даних»

Виконала:

студентка групи КН-308

Ріжко Марія

Перевірила:

Кривенчук Ю.П.

Львів 2021 р.

Тема:

Аналітичні рішення за допомогою нейронних мереж, дерева рішень, на основі самоорганізуючих карт Кохонена, асоціативних правил, трансформації даних і прогнозування за допомогою лінійної регресії.

Аналітичні рішення за допомогою лінійної регресії

Для початку обробимо дані.

Треба заповнити пропущені значення. Оскільки є пропущено лише 434 і 511 значень з 9576, то замінімо пропущені 'engV' середнім, а 'drive' модою.

```
df.isnull().sum()

buyer      0
car        0
price      0
body       0
mileage    0
engV      434
engType    0
registration 0
year       0
model      0
drive     511
dtype: int64

df.shape

(9576, 11)

df['drive'].value_counts()

front     5188
full     2500
rear     1377
Name: drive, dtype: int64

df['engV'].fillna(df['engV'].mean(), inplace=True)
df['drive'].fillna('front', inplace=True)
df.isnull().sum()
```

Модель машини замінімо на 'other', якщо ця модель зустрічається не більше 100 раз в датасеті.

```
mark = df['car'].value_counts() > 100
mark = mark[mark == False]
mark.index
```

```
Index(['Porsche', 'Infiniti', 'Suzuki', 'Smart', 'Geely', 'Chery', 'SsangYong',
      'Seat', 'GAZ', 'Volvo', 'Chrysler', 'Jeep', 'Tesla', 'UAZ', 'Jaguar',
      'Bentley', 'Dodge', 'MINI', 'Acura', 'Dacia', 'Moskvich-AZLK',
      'Alfa Romeo', 'Great Wall', 'BYD', 'Lincoln', 'ËUAZ', 'Moskvich-Izh',
      'Hummer', 'Lifan', 'MG', 'Rover', 'Lancia', 'Daihatsu', 'Cadillac',
      'Aston Martin', 'GMC', 'Isuzu', 'JAC', 'Groz', 'Bogdan', 'Ferrari',
      'Samand', 'Dadi', 'Rolls-Royce', 'Lamborghini', 'Huanghai', 'Mercury',
      'Hafei', 'Barkas', 'Changan', 'Samsung', 'Saab', 'Aro', 'Wartburg',
      'Fisker', 'ZX', 'Maserati', 'Buick', 'TATA', 'SMA', 'FAW',
      'Other-Retro'],
      dtype='object')
```

```
df.loc[df['car'].isin(mark.index), 'car'] = 'other'
df['car'].value_counts()
```

```
Volkswagen      936
Mercedes-Benz   921
other            839
BMW             694
Toyota          541
VAZ             489
Renault         469
Audi            457
Opel            400
Skoda           368
Nissan           368
Hyundai         367
Ford            350
Mitsubishi      327
Chevrolet       246
Daewoo          235
Kia             215
Honda           206
Mazda           198
Peugeot         182
Lexus           174
Land Rover      151
Fiat            119
Subaru          114
Citroen         108
ZAZ             102
Name: car, dtype: int64
```

Колонку моделі видалимо, оскільки вона має 888 унікальних категоріальних значень.

```
df['model'].value_counts()
```

```
E-Class      199
A6            172
Camry         134
Vito iãññ.    131
Lanos         127
...
```

```
340          1
Ideal         1
ML 550        1
Bipper iãññ.  1
M35           1
```

```
Name: model, Length: 888, dtype: int64
```

```
df.drop('model', axis=1, inplace=True)
df.head()
```

	buyer	car	price	body	mileage	engV	engType	registration	year	drive
0	1964	Ford	15500.0	crossover	68	2.500000	Gas	yes	2010	full
1	460	Mercedes-Benz	20500.0	sedan	173	1.800000	Gas	yes	2011	rear
2	1283	Mercedes-Benz	35000.0	other	135	5.500000	Petrol	yes	2008	rear
3	604	Mercedes-Benz	17800.0	van	162	1.800000	Diesel	yes	2012	front
4	298	Mercedes-Benz	33000.0	vagon	91	2.646344	Other	yes	2013	front

Колонку покупця видалимо, оскільки вона має 1982 унікальних категоріальних значень.

```
df['buyer'].value_counts()
```

```
412      15
1311     14
1979     13
907      13
355      13
..
362       1
872       1
1148      1
1132      1
1045      1
Name: buyer, Length: 1982, dtype: int64
```

```
df.drop('buyer', axis=1, inplace=True)
df.head()
```

	car	price	body	mileage	engV	engType	registration	year	drive
0	Ford	15500.0	crossover	68	2.500000	Gas	yes	2010	full
1	Mercedes-Benz	20500.0	sedan	173	1.800000	Gas	yes	2011	rear
2	Mercedes-Benz	35000.0	other	135	5.500000	Petrol	yes	2008	rear
3	Mercedes-Benz	17800.0	van	162	1.800000	Diesel	yes	2012	front
4	Mercedes-Benz	33000.0	vagon	91	2.646344	Other	yes	2013	front

Перетворимо категоріальні значення в дискретні.

```
df = pd.get_dummies(df)
df.head()
```

	price	mileage	engV	year	car_Audi	car_BMW	car_Chevrolet	car_Citroen	car_Daewoo	car_Fiat	...	body_van	engType_Diesel	engType_Gas	engType_Other
0	15500.0	68	2.500000	2010	0	0	0	0	0	0	...	0	0	1	0
1	20500.0	173	1.800000	2011	0	0	0	0	0	0	...	0	0	1	0
2	35000.0	135	5.500000	2008	0	0	0	0	0	0	...	0	0	0	0
3	17800.0	162	1.800000	2012	0	0	0	0	0	0	...	1	1	0	0
4	33000.0	91	2.646344	2013	0	0	0	0	0	0	...	0	0	0	1

Розділимо дані на тест і трейн. Тренуємо лінійну регресію. Візуалізуємо результати. Виведемо помилки.

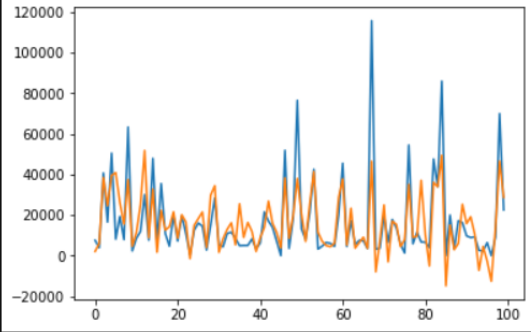
```
X_train, X_test, y_train, y_test = train_test_split(df.drop('price', axis=1), df['price'], test_size=0.3, random_state=42)

reg = LinearRegression().fit(X_train, y_train)

y_res = reg.predict(X_test)

plt.plot(range(100), y_test[:100])
plt.plot(range(100), y_res[:100])

[<matplotlib.lines.Line2D at 0x20eabb1d3d0>]
```



```
reg.score(X_test, y_test)

0.3287267514691038

mean_squared_error(y_test, y_res)

421760853.28926766
```

Початкова помилка $MSE = 421760853$ і $R = 0.328$. Від неї будемо відштовхуватись у наступних моделях.

Аналітичні рішення за допомогою дерева рішень

Будуємо дерева рішень з глибиною від 1 до 15.

```
errors = []
for i in range(1, 15):
    dt = DecisionTreeRegressor(max_depth=i)
    dt.fit(X_train, y_train)
    y_res = dt.predict(X_test)
    errors.append(mean_squared_error(y_test, y_res))
```

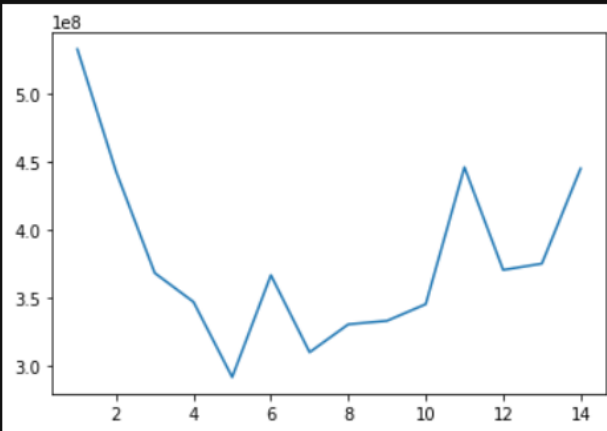
errors

```
[532695810.55537754,
442911177.57442355,
368338908.307972,
347149687.2825448,
291867592.18748087,
366780932.30739814,
310143294.22083,
330687546.3282044,
333254630.53718615,
345482056.17661464,
446027953.55590934,
370620156.73257035,
375253361.4278655,
445062247.1031801]
```

Виведемо результати на графік.

```
plt.plot(range(1, 15), errors)
```

```
[<matplotlib.lines.Line2D at 0x20ead13da30>]
```

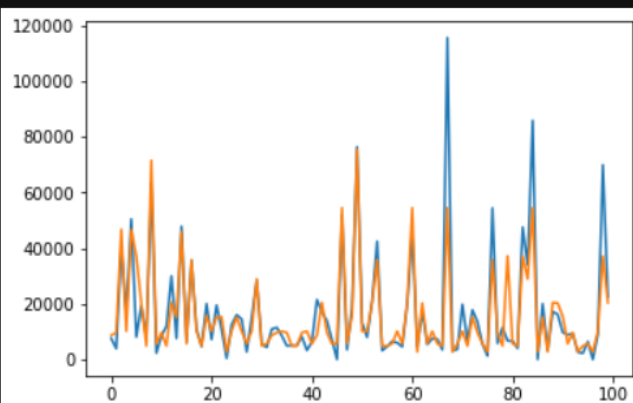


Найкращий результат показало дерево з глибиною 5 і новий MSE = 291867592 і $R = 0.506$

Візуалізуємо результати.

```
dt = DecisionTreeRegressor(max_depth=5)
dt.fit(X_train, y_train)
y_res = dt.predict(X_test)
plt.plot(range(100), y_test[:100])
plt.plot(range(100), y_res[:100])
```

```
[<matplotlib.lines.Line2D at 0x20ea73b7790>]
```



```
dt.score(X_test, y_test)
```

```
0.5354644575934457
```

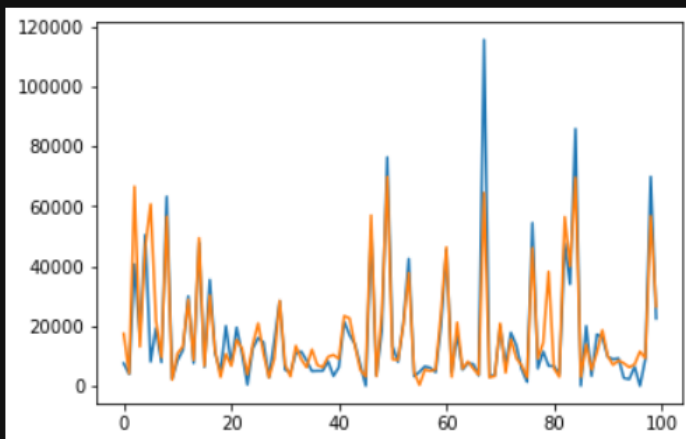
Аналітичні рішення за допомогою нейронних мереж

```
reg = MLPRegressor(random_state=1, max_iter=10000)
reg.fit(X_train, y_train)
```

```
MLPRegressor(max_iter=10000, random_state=1)
```

```
y_res = reg.predict(X_test)
plt.plot(range(100), y_test[:100])
plt.plot(range(100), y_res[:100])
```

```
[<matplotlib.lines.Line2D at 0x20ea87e61f0>]
```



```
mean_squared_error(y_test, y_res)
```

```
270486218.1850499
```

```
reg.score(X_test, y_test)
```

```
0.5694949852555803
```

Побудовано багатошаровий перцептрон з стандартними характеристиками `hidden_layer_sizes=100`, функція активації `relu`, вирішувач для оптимізації ваг `'adam'`, максимальна кількість ітерацій встановлена 10000.

$MSE = 270486218$ і $R = 0.569$

Цей результат кращий за лінійну регресію та дерево рішень.

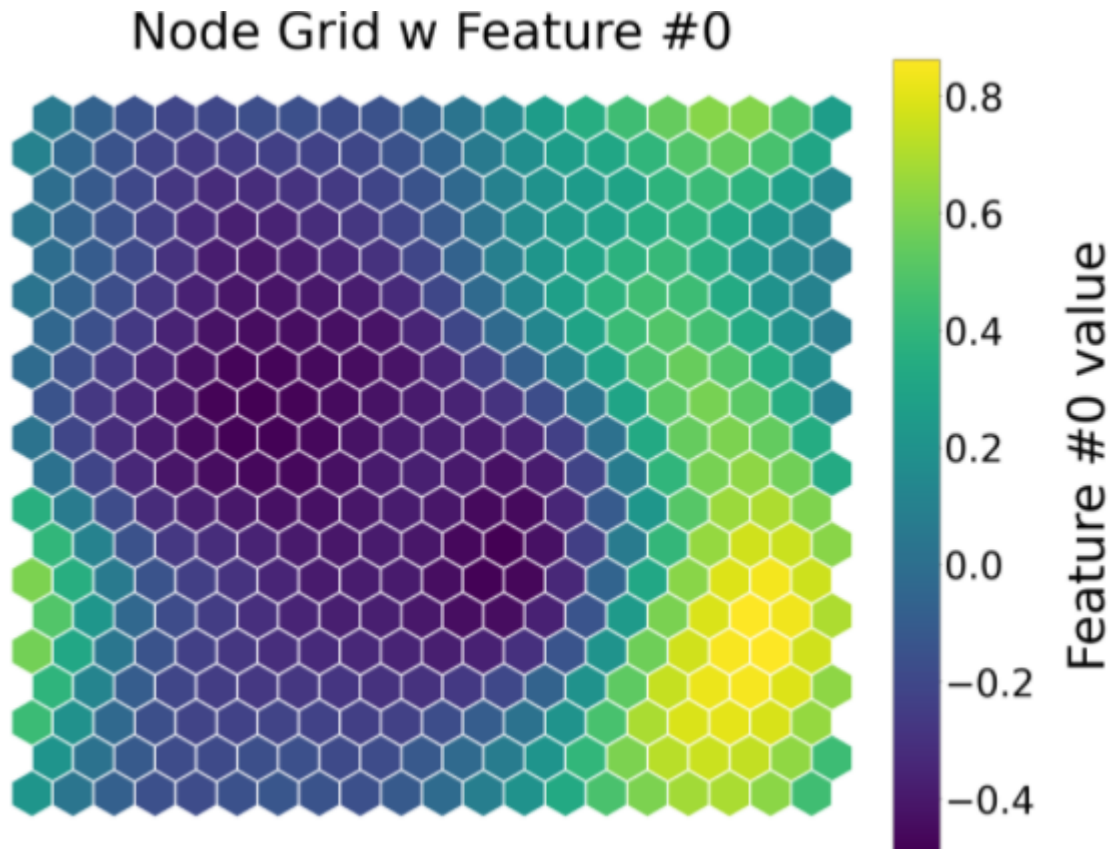
Аналітичні рішення на основі самоорганізуючих карт Кохонена

Побудуємо мережу 20×20 і натренуємо її на 20000 епохи з початковою швидкістю навчання 0.01


```
net = sps.somNet(20, 20, X_train, PBC=True)
net.train(0.01, 20000)
```

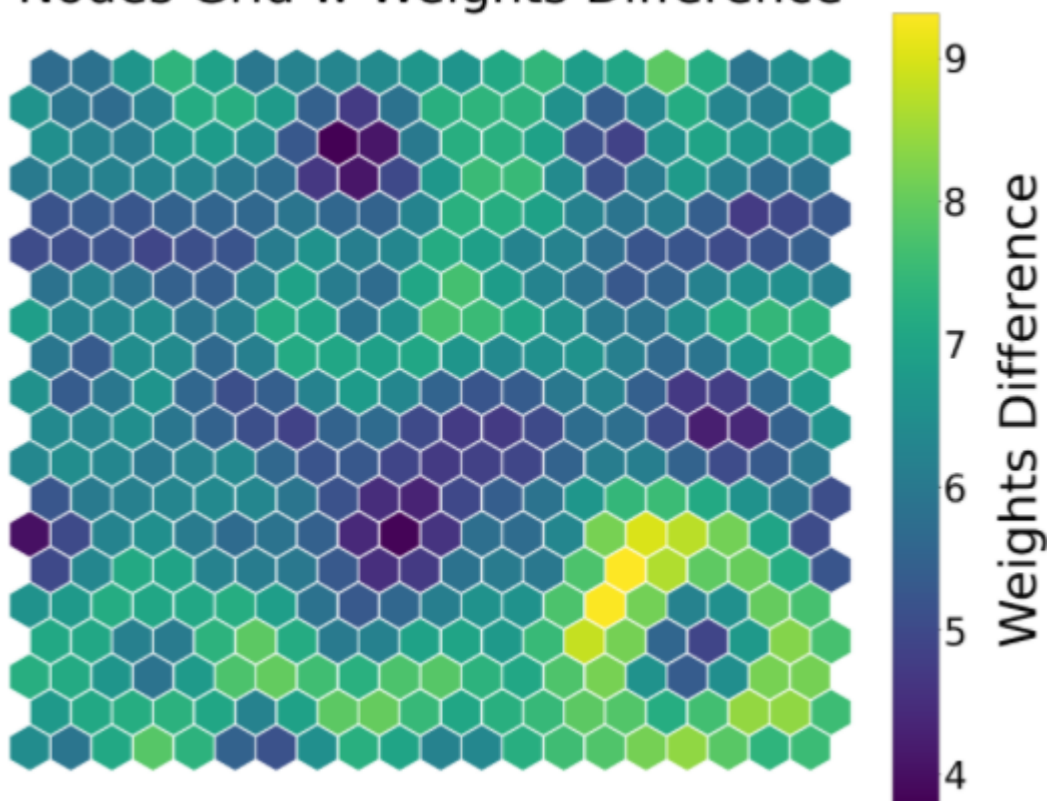
Periodic Boundary Conditions active.
The weights will be initialised randomly.
Training SOM... done!

Зобразимо мапу мережеві і змалюємо їх відносно першої фічі датасету.



Зобразимо мапу мережі і змалюємо її відносно відстаней між кожною вершиною та її сусідами.

Nodes Grid w Weights Difference



Аналітичні рішення на основі асоціативних правил

Завантажимо новий датасет, зчитаємо його.

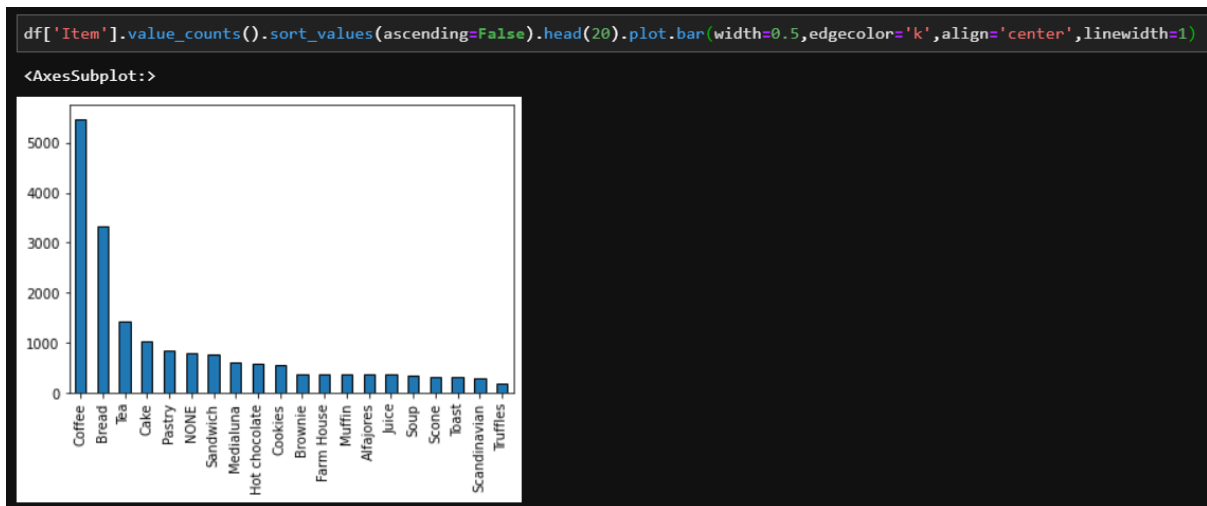
```
df = pd.read_csv('BreadBasket_DMS.csv')
df.head()
```

	Date	Time	Transaction	Item
0	2016-10-30	09:58:11	1	Bread
1	2016-10-30	10:05:34	2	Scandinavian
2	2016-10-30	10:05:34	2	Scandinavian
3	2016-10-30	10:07:57	3	Hot chocolate
4	2016-10-30	10:07:57	3	Jam

```
df['Item'].value_counts()
```

```
Coffee      5471
Bread       3325
Tea         1435
Cake        1025
Pastry       856
...
Raw bars      1
The BART      1
Gift voucher  1
Chicken sand  1
Adjustment    1
Name: Item, Length: 95, dtype: int64
```

Візуалізуємо топ-20 проданих товарів.



Перетворюємо датасет на потрібний формат.

```
hot_encoded_df = df.groupby(['Transaction', 'Item'])['Item'].count().unstack().reset_index().fillna(0).set_index('Transaction')
hot_encoded_df.head()
```

Item	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Truffles	Tshirt	Valentin cz
Transaction																		
1	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
2	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
3	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	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
5	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

```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

df = df.applymap(encode_units)
df.head()
```

Item	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	Toast	Truffles	Tshirt	Valentin cz
Transaction																		
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

Проведемо асоціативний аналіз.

```
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
rules.head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Alfajores)	(Coffee)	0.036093	0.475081	0.019515	0.540698	1.138116	0.002368	1.142861
1	(Coffee)	(Alfajores)	0.475081	0.036093	0.019515	0.041078	1.138116	0.002368	1.005199
2	(Pastry)	(Bread)	0.085510	0.324940	0.028958	0.338650	1.042194	0.001172	1.020731
3	(Bread)	(Pastry)	0.324940	0.085510	0.028958	0.089119	1.042194	0.001172	1.003961
4	(Brownie)	(Coffee)	0.039765	0.475081	0.019515	0.490765	1.033013	0.000624	1.030799
5	(Coffee)	(Brownie)	0.475081	0.039765	0.019515	0.041078	1.033013	0.000624	1.001369
6	(Coffee)	(Cake)	0.475081	0.103137	0.054349	0.114399	1.109196	0.005350	1.012717
7	(Cake)	(Coffee)	0.103137	0.475081	0.054349	0.526958	1.109196	0.005350	1.109667
8	(Hot chocolate)	(Cake)	0.057916	0.103137	0.011331	0.195652	1.897010	0.005358	1.115019
9	(Cake)	(Hot chocolate)	0.103137	0.057916	0.011331	0.109868	1.897010	0.005358	1.058364

Support це відсоток транзакцій, що містять певну комбінацію елементів відносно загальної кількості транзакцій у базі даних.

Confidence вимірює, наскільки наслідковий елемент залежить від апіорного елементу.

Lift - це міра подолання проблеми з підтримкою та впевненістю. Він вимірює різницю між достовірністю правила та очікуваною впевненістю.

Подивимось на асоціативні правила, де Confidence більше 0.5 та lift > 1.

```
rules[(rules['lift'] >= 1) & (rules['confidence'] >= 0.5)]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Alfajores)	(Coffee)	0.036093	0.475081	0.019515	0.540698	1.138116	0.002368	1.142861
7	(Cake)	(Coffee)	0.103137	0.475081	0.054349	0.526958	1.109196	0.005350	1.109667
12	(Cookies)	(Coffee)	0.054034	0.475081	0.028014	0.518447	1.091280	0.002343	1.090053
15	(Hot chocolate)	(Coffee)	0.057916	0.475081	0.029378	0.507246	1.067704	0.001863	1.065276
17	(Juice)	(Coffee)	0.038296	0.475081	0.020460	0.534247	1.124537	0.002266	1.127031
18	(Medialuna)	(Coffee)	0.061379	0.475081	0.034939	0.569231	1.198175	0.005779	1.218561
22	(NONE)	(Coffee)	0.079005	0.475081	0.042073	0.532537	1.120938	0.004539	1.122908
24	(Pastry)	(Coffee)	0.085510	0.475081	0.047214	0.552147	1.162216	0.006590	1.172079
26	(Sandwich)	(Coffee)	0.071346	0.475081	0.037981	0.532353	1.120551	0.004086	1.122468
28	(Scone)	(Coffee)	0.034309	0.475081	0.017941	0.522936	1.100729	0.001642	1.100310
30	(Spanish Brunch)	(Coffee)	0.018046	0.475081	0.010807	0.598837	1.260494	0.002233	1.308493
33	(Toast)	(Coffee)	0.033365	0.475081	0.023502	0.704403	1.482699	0.007651	1.775789

Висновок

За час виконання даної лабораторної роботи я дослідила аналітичні рішення за допомогою нейронних мереж, дерева рішень, на основі самоорганізуючих карт Кохонена, асоціативних правил, трансформації даних і прогнозування за допомогою лінійної регресії.