Multi-Objective Recommender Systems

Presented by Mark Rosado and Samar Ashrafi
At Wentworth Institute of Technology
Fall 2022
Data Science course. Prof. Memo Ergezer.

Presentation Outline

1-Background

2-Initial Findings

3-Preprocessing/Cleaning the Data

4-Modeling

5-Findings

6-Conclusion

Introduction of Multi-Objective Recommender:

A competition hosted by OTTO, famous online-shopping in Germany

- Goal:Predict e-commerce Clicks,cart addition, and orders
  (predict aid for each session)

	events
session	
0	[{'aid': 1517085, 'ts': 1659304800025, 'type':...
1	[{'aid': 424964, 'ts': 1659304800025, 'type': ...
2	[{'aid': 763743, 'ts': 1659304800038, 'type': ...
3	[{'aid': 1425967, 'ts': 1659304800095, 'type':...
4	[{'aid': 613619, 'ts': 1659304800119, 'type': ...
...	...
99995	[{'aid': 1387489, 'ts': 1659326711310, 'type':...
99996	[{'aid': 1091948, 'ts': 1659326711396, 'type':...
99997	[{'aid': 366639, 'ts': 1659326711431, 'type': ...
99998	[{'aid': 845181, 'ts': 1659326711611, 'type': ...
99999	[{'aid': 601639, 'ts': 1659326711757, 'type': ...

```
[{'aid': 613619, 'ts': 1659304800119, 'type': 'clicks'},
 {'aid': 298827, 'ts': 1659304836708, 'type': 'clicks'},
 {'aid': 298827, 'ts': 1659304900468, 'type': 'orders'},
 {'aid': 383828, 'ts': 1661161611985, 'type': 'clicks'},
 {'aid': 255379, 'ts': 1661161636464, 'type': 'clicks'},
 {'aid': 1838173, 'ts': 1661161670830, 'type': 'clicks'},
 {'aid': 1453726, 'ts': 1661161695814, 'type': 'clicks'},
 {'aid': 1838173, 'ts': 1661161708717, 'type': 'clicks'},
 {'aid': 255379, 'ts': 1661161751223, 'type': 'clicks'},
 {'aid': 383828, 'ts': 1661161753524, 'type': 'clicks'},
 {'aid': 1554752, 'ts': 1661504170116, 'type': 'clicks'},
 {'aid': 1554752, 'ts': 1661504180466, 'type': 'carts'},
```


Click
2022/08/01 07:00:36

Order
2022/08/01 07:01:40

Click
2022/08/26 17:56:10

Add into a Cart
2022/08/26 17:56:20

Initial Findings

Data

- The data that is given to us contains more than 5,000,000 observations.
    - This became impossible to run the entire file at once with the devices we had.
- The kaggle challenge provided multiple discussion topics and tips about the competition and anything related to it.
    - We utilized a forum called "[OTTO] Easy understanding for beginners."

First Thoughts

- We first utilized the Kaggle IDE but it is very limited compared to other IDEs.
    - We then switched onto Jupyter which provided better results.

PreProcessing / Cleaning Data

- Converting json format to Pandas DataFrame

- No Nan values to delete

```
df.isna().sum()
```

```
session    0
aid        0
ts         0
type       0
dtype: int64
```

Convert categorical Data to numeric Data

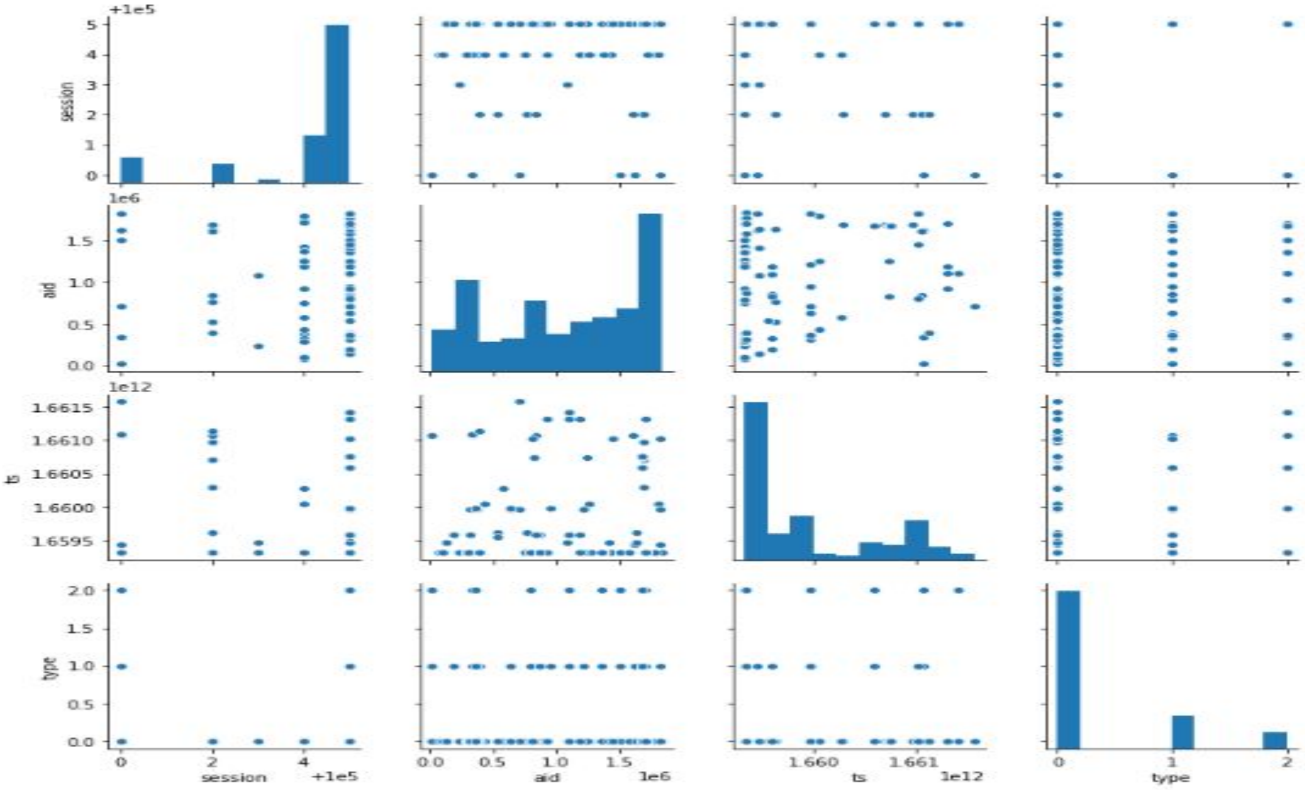
df.replace({'clicks': 0, 'carts': 1,'orders':2}, inplace=True)

	session	aid	ts	type
0	100000	1498214	1659326712113	0
1	100000	1617298	1659445460457	0
2	100000	1617298	1659445471474	1
3	100000	1820189	1659445496027	0
4	100000	1619534	1661072158119	0
5	100000	22770	1661076416668	0
6	100000	22770	1661076452638	0
7	100000	22770	1661076458938	1
8	100000	339965	1661076485171	0
9	100000	339965	1661076499914	1

- **Plot the data to select best features:**

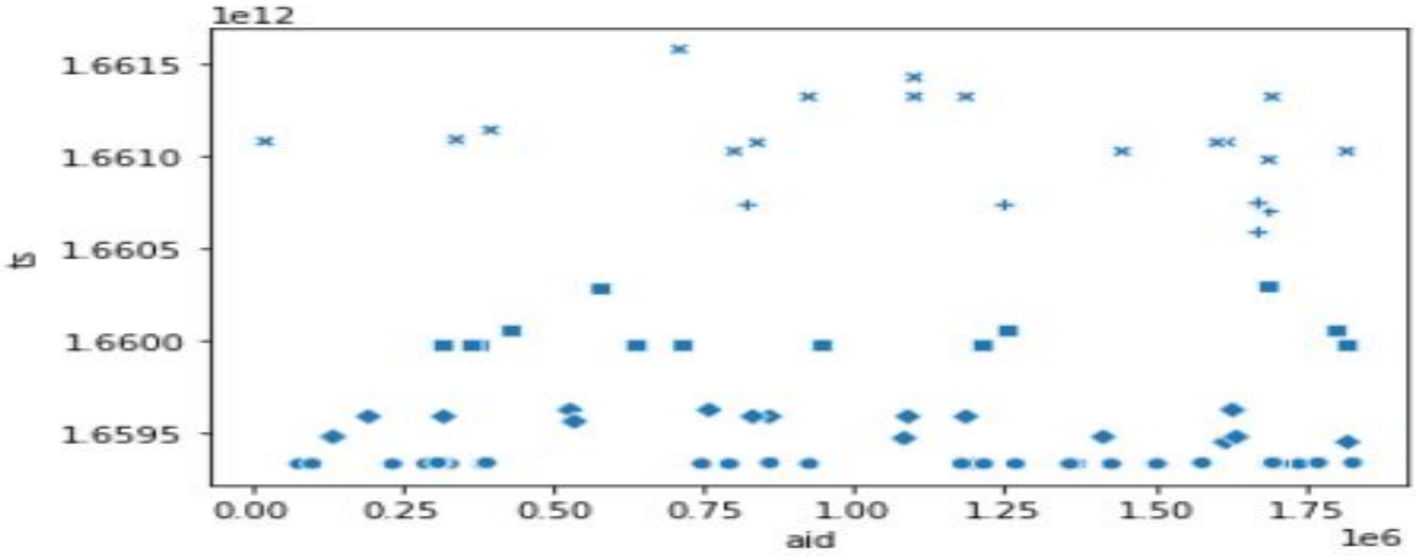
```
sns.pairplot(newData[['session', 'aid', 'ts', 'type']])
```

```
<seaborn.axisgrid.PairGrid at 0x7fcb957c6c50>
```



```
# In[10]:
print(newData['Clusters'].value_counts() )
#plot
sns.scatterplot(x="aid", y="ts", style="Clusters", legend = False, data=newData);
```

```
0    64
1    27
2    25
4    20
3    15
Name: Clusters, dtype: int64
```



Modeling

4 different Machine Learning algorithms were used:

- Supervised
    - K-Nearest Neighbors
    - Logistic Regression
    - Decision Trees
- Unsupervised
    - K-means Clustering

Small samples were originally used so program would run quicker

Models were then chosen based on accuracy score to compare between others

K-Means with aid,ts

```
# K- means Clustering
num_clusters=5
kmeans = cluster.KMeans(n_clusters=num_clusters, init="k-means++")
kmeans = kmeans.fit(newData[['aid', 'ts']])
kmeans.cluster_centers_
## Attach Cluster to Original Data
newData['Clusters'] = kmeans.labels_
#predict up to 20 values of aid for each session
for i in range(5):
    f=newData[newData['Clusters']==i]['aid']
    f = f.iloc[:20].to_string(index=False)
    f=f.strip()
    f=f.replace('\n', ',')
    newData.loc[newData.Clusters==i, 'Predictions'] = f
newData.head(10)
```

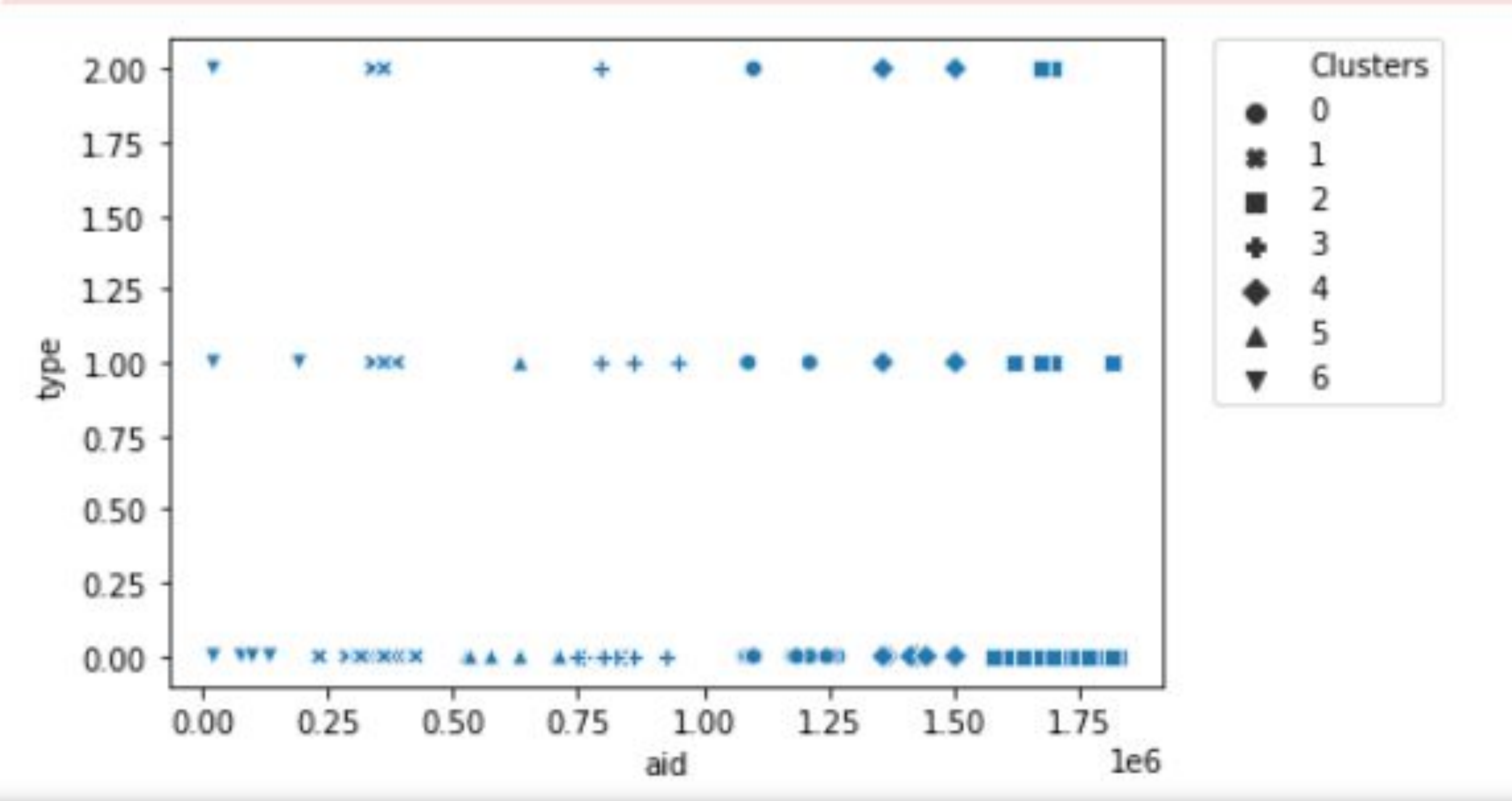
K-Means with aid,ts Results

	session	aid	ts	type	Clusters	Predictions
0	100000	1498214	1659326712113	0	0	1498214,1689118, 234322, 383036,1200197, 28934...
1	100000	1617298	1659445460457	0	4	1617298,1617298,1820189, 759998, 529006,162909...
2	100000	1617298	1659445471474	1	4	1617298,1617298,1820189, 759998, 529006,162909...
3	100000	1820189	1659445496027	0	4	1617298,1617298,1820189, 759998, 529006,162909...
4	100000	1619534	1661072158119	0	1	1619534, 22770, 22770, 22770, 339965, 33996...
5	100000	22770	1661076416668	0	1	1619534, 22770, 22770, 22770, 339965, 33996...
6	100000	22770	1661076452638	0	1	1619534, 22770, 22770, 22770, 339965, 33996...
7	100000	22770	1661076458938	1	1	1619534, 22770, 22770, 22770, 339965, 33996...
8	100000	339965	1661076485171	0	1	1619534, 22770, 22770, 22770, 339965, 33996...
9	100000	339965	1661076499914	1	1	1619534, 22770, 22770, 22770, 339965, 33996...

K-Means with aid,type

```
#K- means Clustering for aid and type
kmeans = cluster.KMeans(n_clusters=7, init="k-means++")
kmeans = kmeans.fit(newData[['aid', 'type']], newData[['session']])
kmeans.cluster_centers_
## Attach Cluster to Original Data
newData['Clusters'] = kmeans.labels_
newData.head(10)

newData['Clusters'].value_counts()
#plot
sns.scatterplot(x="aid", y="type", style="Clusters", data=newData)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0);
```

K-Means with aid,type Results



K-Means with All Feautures + additional feature

	session	aid	ts	type	minutes
0	0	1517085	1659304800025	0	1.741433
1	0	1563459	1659304904511	0	1042.248583
2	0	1309446	1659367439426	0	4.676183
3	0	16246	1659367719997	0	2.522450
4	0	1781822	1659367871344	0	0.240867
...	...	...	...	...	...
10676	99	369914	1661390037098	0	5383.902217
10677	99	759787	1661713071231	0	1.222433
10678	99	759787	1661713144577	0	0.549300
10679	99	1400630	1661713177535	0	1.119533

```python
ii = IterativeImputer()
ii.fit(train_df)
train_df = ii.transform(train_df)
test_df = ii.transform(test_df)
num_clusters=train_df.shape[0]//20
kmeans = cluster.KMeans(n_clusters=num_clusters, init='random',
    max_iter=300,
    tol=1e-04, random_state=42)
kmeans.fit(train_df)
predictions  = kmeans.predict(train_df)
kmeans.cluster_centers_
kmeans.labels_
```

```
array([ 80,  80,  73, ..., 272, 272, 272], dtype=int32)
```

```
array([[8.45454545e+01, 3.15775273e+05, 1.65969990e+12, 1.38777878e-17,
        3.51804848e+00],
       [5.01538462e+01, 3.15008308e+05, 1.65930523e+12, 1.53846154e-01,
        2.30071032e+01],
       [6.89047619e+01, 1.14072667e+06, 1.65930496e+12, 2.38095238e-01,
        8.34497246e+01],
       ...,
       [3.82000000e+01, 3.18168920e+05, 1.66025480e+12, 8.00000000e-02,
        6.43874667e-01],
       [4.44375000e+01, 8.21055750e+05, 1.65965126e+12, 0.00000000e+00,
        4.12926734e+02],
       [1.40000000e+01, 3.64860400e+05, 1.65956969e+12, 0.00000000e+00,
        5.29550000e-01]])
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
X = newData[['ts', 'type']]
y = newData[['aid']]
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)
y_pred = knn.predict(X)
print(metrics.accuracy_score(y, y_pred))
```

```
0.5099337748344371
```

KNN

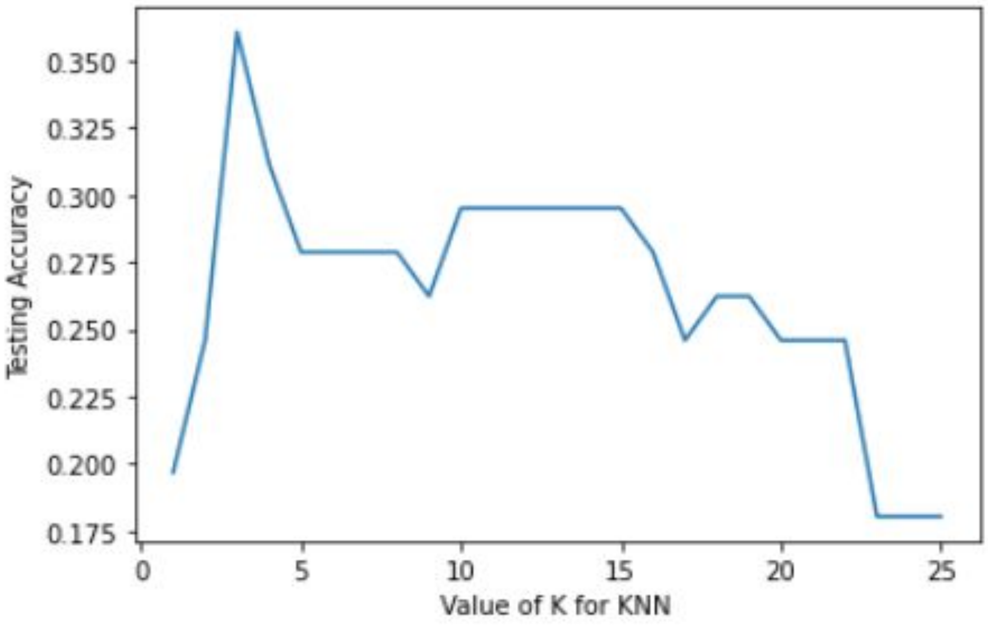
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.2786885245901639
0.19672131147540983
```

Select the best K

```
plt.plot(k_range, scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
```

```
Text(0, 0.5, 'Testing Accuracy')
```



Logistic Regression

```
# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=4)
```

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/utils/validation.py:760: DataConversionWarning: A colum
n a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# STEP 3: make predictions on the testing set
y_pred = logreg.predict(X_test)
```

```
# compare actual response values (y_test) with predicted response values (y_pred)
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.06557377049180328
```

Decision Tree

```
dtc = DecisionTreeClassifier(min_samples_split=5, random_state=0)

dtc.fit(X_train, y_train)

y_pred_class = dtc.predict(X_test)
metrics.accuracy_score(y_test, y_pred_class)
```

```
0.3114754098360656
```

Conclusion:

Ending Thoughts

- Overall, KNN was the best model in terms of error rate with testing data
- We can probably receive a better model if we are able to manipulate existing features
- Due to the fact that the dataset was large we couldn't properly include every data point but there might be ways to work with big data that we don't know at the moment

Thank you!

Questions and Comments?